OPTIMIZATION-BASED METHODS FOR DETERMINISTIC AND STOCHASTIC CONTROL: ALGORITHMIC DEVELOPMENT, ANALYSIS AND APPLICATIONS ON MECHANICAL SYSTEMS & FIELDS

A Dissertation
Presented to
The Academic Faculty

By

Georgios Boutselis

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Aerospace Engineering

Georgia Institute of Technology

May 2020

OPTIMIZATION-BASED METHODS FOR DETERMINISTIC AND STOCHASTIC CONTROL: ALGORITHMIC DEVELOPMENT, ANALYSIS AND APPLICATIONS ON MECHANICAL SYSTEMS & FIELDS

Approved by:

Dr. Evangelos Theodorou, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Melvin Leok
Department of Mathematics
University of California, San Diego

Dr. Andrzej Swiech
School of Mathematics
Georgia Institute of Technology

Dr. Efstathios Bakolas
Department of Aerospace Engineering and Engineering Mechanics
The University of Texas at Austin

Dr. Yongxin Chen
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: December 9, 2019

To Evri and Penny.

ACKNOWLEDGEMENTS

Foremost, I thank my advisor Prof. Evangelos Theodorou for giving me the freedom to explore different research areas and putting his faith in my ideas. I am ingratiated for his confidence in me, introducing me to basic concepts in control and machine learning in the initial period of my studies and letting me work independently in the latter. I also found great value in discussions I had with professor Theodorou about academia and doctoral studies in general, as well as important values that should shape each human being.

I am very grateful that I had the opportunity to interact with Prof. Melvin Leok and Prof. Andrzej Swiech about certain topics in my work. Their experience and insights created a stimulating environment, helping me elevate my research agenda and think about important extensions. I also thank Prof. Yongxin Chen and Prof. Efstathios Bakolas for their informed comments and suggestions on my thesis topic.

A very important part of my PhD journey was a unique set of labmates and friends, to whom I owe a large number of positive experiences. I feel very lucky to have spent time with such a great group of individuals with diverse backgrounds, who always seemed up for an interesting and pleasant discussion: Orestis Vasios, Marcus Pereira, Keuntaek Lee, Aoyama Yuichiro, Ethan Evans, Gabriel Nakajima, Kaivalya Bakshi, Pat Wang, Emre Yilmaz, David Fan, Yunpeng Pan, Manan Gandhi, Kamil Saigol, Grady Williams, Camilo Duarte, Emily Reed, Tianrong Chen, Bogdan Vlahov, Jintasit Pravitra, Akash Patel, Hassan Almubarak, Guan Liu, Rahul Singh and Jason Gibson.

Last but not least, I would have never been able to complete my doctoral studies without the support and love of my family. Their understanding and constant sacrifices have been remarkable. Finally, I express my gratitude to A.S. Onassis Foundation for their financial support during all these years, which allowed me to concentrate on my work.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

Developing efficient control algorithms for practical scenarios remains a key challenge for the scientific community. Towards this goal, optimal control theory has been widely employed over the past decades, with applications both in simulated and real environments. Unfortunately, standard model-based approaches become highly ineffective when modeling accuracy degrades. This may stem from erroneous estimates of physical parameters (e.g., friction coefficients, moments of inertia), or dynamics components which are inherently hard to model. System uncertainty should therefore be properly handled within control methodologies for both theoretical and practical purposes. Of equal importance are state and control constraints, which must be effectively handled for safety critical systems.

To proceed, the majority of works in controls and reinforcement learning literature deals with systems lying in finite-dimensional Euclidean spaces. For many interesting applications in aerospace engineering, robotics and physics, however, we must often consider dynamics with more challenging configuration spaces. These include systems evolving on differentiable manifolds, as well as systems described by stochastic partial differential equations. Some problem examples of the former case are spacecraft attitude control, modeling of elastic beams and control of quantum spin systems. Regarding the latter, we have control of thermal/fluid flows, chemical reactors and advanced batteries.

This work attempts to address the challenges mentioned above. We will develop numerical optimal control methods that explicitly incorporate modeling uncertainty, as well as deterministic and probabilistic constraints into prediction and decision making. Our iterative schemes provide scalability by relying on dynamic programming principles as well as sampling-based techniques. Depending upon different problem setups, we will handle uncertainty by employing suitable concepts from machine learning and uncertainty quantification theory. Moreover, we will show that well-known numerical control methods can be extended for mechanical systems evolving on manifolds, and dynamics described by

stochastic partial differential equations. Our algorithmic derivations utilize key concepts from optimal control and optimization theory, and in some cases, theoretical results will be provided on the convergence properties of the proposed methods. The effectiveness and applicability of our approach are highlighted by substantial numerical results on simulated test cases.

CHAPTER 1
INTRODUCTION

1.1 Motivation and Prior Work

The main objective of this proposal is the development of unconstrained and constrained, optimization-based control algorithms for deterministic and stochastic/probabilistic systems. One key question we ask is how *uncertainty* can be properly incorporated into control and trajectory-optimization methodologies. In general, stochastic systems can be modeled in various ways; some approaches treat dynamics as stochastic differential equations [1], while others rely on probabilistic inference techniques from Machine Learning (ML) [2]. We will work with different uncertainty representation approaches, each being suitable for a particular set of applications. These will be combined with well-established optimal control methods, namely Differential Dynamic Programming (DDP) [3] and Path Integral control (PI) [1], which avoid the *curse-of-dimensionality* issue through local approximations and sampling techniques, respectively.

We will first consider the problem of controlling systems with uncertain internal parameters and initial states. Towards this goal, we will quantify uncertainty by utilizing Polynomial Chaos theory. Wiener introduced Polynomial Chaos [4], and used it to decompose stochastic processes into a convergent series of Hermite polynomials. Xiu and Karniadakis in [5] extended this to the generalized Polynomial Chaos (gPC) framework by employing orthogonal polynomials from the Askey-scheme. Our developed control scheme, called gPC-DDP, relies on dynamic programming principles and is capable of controlling the probabilistic behavior of the trajectory in an optimization setting. When mechanical systems are considered, the performance of the proposed methodology is further improved by incorporating the concept of Variational Integrators (VI's). VI's are a class of numerical

time stepping methods derived from discretization of Hamilton's principle [6, 7]. They have been shown to outperform standard numerical integration schemes (e.g., Euler or RK) due to their long-term energy preserving properties. We will develop variational integration schemes for gPC-based dynamics, and incorporate them in our control methodology.

Next, we will attempt to learn control tasks, when data from the agent and its environment are accessible. While model-free reinforcement learning methods have demonstrated impressive results in robotic control applications [8], they typically require human expert demonstrations and a relatively large number of interactions with the physical system. We propose a model-based approach to address the issue of sample inefficiency by learning explicit transition dynamics models from data. Our probabilistic trajectory optimizer extends DDP by incorporating Guassian Process (GP) regression methods. GPs are used to define probability distributions over continuous functions and offer a powerful way to perform Bayesian nonparametric estimation of functions [9]. The developed algorithm, called Probabilistic Differential Dynamic Programming (PDDP), presents efficiency both in computational resources and sample size, while avoiding explicit policy parameterizations. We would also like to mention that convergence properties for gPC-DDP and PDDP are provided which generalize previous theoretical work on DDP-related methods.

We will extend the above techniques to handle state and control constraints. Regarding linearization-based control, we will show how Lagrangian methods and KKT optimality conditions [10] can be incorporated within the standard DDP algorithm. The former will require a specific use of penalty functions, while the latter will rely on a modification of Bellman's optimality principle. We will show how a combination of the above approaches numerically outperforms standard off-the-shelf methods and previous works for constrained trajectory optimization. These ideas will subsequently be used within the gPC and GP frameworks to allow for constrained probabilistic control. Besides the aforementioned approaches, a sampling-based trajectory optimization scheme will also be developed via stochastic approximation theory [11] for constrained discrete dynamical systems. We will

show that the obtained algorithm generalizes the Path Integral control framework. Moreover, this approach will allow us to used non-smooth penalty methods to derive sampling, constrained control algorithms, which will be tested in simulation and will be compared against previous works.

The research discussed above has a major limitation: It is only applicable to finite-dimensional Euclidean spaces. However, real physical systems often admit more complex configuration spaces. It can be shown, for example, that the set of rigid body transformations behaves as a differentiable manifold [12]. Moreover, stochastic partial differential equations (SPDEs) have been used to model diverse applications ranging from turbulence to plasma physics [13]. The remaining parts of this thesis will deal with these aforementioned cases. Regarding the former, we will formulate a Lie-theoretic version of the DDP and PI control algorithms for systems evolving on smooth manifolds, and cover topics spanning their derivation, convergence analysis and numerical behavior. For the latter case, we will develop a sampling-based framework for control of stochastic fields. Departing from finite-dimensional representations, we show that optimal control of SPDEs can be casted as a variational optimization problem and therefore be solved using direct sampling of infinite-dimensional processes. We will find that such a framework can optimize over open-loop, as well as arbitrary (parameterized) feedback policies, and its applicability will be tested on simulated PDEs.

1.2 Structure of thesis

The rest of this thesis is organized as follows:

Chapter II: We will give some background on optimal control theory, constrained optimization and barrier certificates methods, which will be used for the development of our unconstrained/constrained trajectory optimizers. We will also include the basics on GPs and gPC theory, as well as preliminaries on SPDEs, discrete mechanics and Lie groups theory.

Chapter III: We will present our results on linearization-based control. We will start by introducing our constrained version of DDP, which will later be incorporated within the GP and gPC formulations to handle parametric/non-parametric uncertainties. A special formulation will be included on discrete mechanical systems with parametric uncertainties, which will make use of Variational Integrators on gPC-based dynamics. This chapter concludes by discussing the geometric formulation of DDP for deterministic systems evolving on manifolds. An extended convergence analysis of the above algorithms will also be presented.

Chapter IV: This chapter will develop sampling-based algorithms for stochastic systems. We will focus on different (but parallel) derivation techniques, which will result in sampling methodologies for control. One main direction lies in Hamilton-Jacobi-Bellman theory, as well as Variational Optimization. Based on these frameworks we will derive control methodologies for Stochastic Partial Differential Equations, as well as systems lying on matrix Lie groups. The parallel approach will use stochastic approximation methods for discrete stochastic systems, and will allow us to handle state and control constraints by optimizing over parameterized sampling distributions.

Chapter IV: This chapter concludes the thesis and discusses possible extensions.

1.3 Publications

The line of research discussed in this thesis resulted in the following articles:

Published Conference papers:

- George I. Boutselis, Gerardo De La Torre and Evangelos A. Theodorou, "Stochastic optimal control using polynomial chaos variational integrators", IEEE American Control Conference (ACC), 2016.

- George I. Boutselis and Evangelos A. Theodorou, "Spectral variational integrators for trajectory optimization under parametric uncertainties and stochastic disturbances",

IEEE Conference on Decision and Control (CDC), 2016.

- Evangelos Theodorou, George I. Boutselis and Kaivalya Bakshi, "Linearly solvable stochastic optimal control for infinite-dimensional systems", IEEE Conference on Decision and Control (CDC), 2018.

- George I. Boutselis and Evangelos Theodorou, "Path integral control on Lie groups", IEEE Conference on Decision and Control (CDC), 2018.

- E. Evans, M. A. Pereira, George I. Boutselis, E. A. Theodorou, "Variational Optimization Based Reinforcement Learning for Infinite Dimensional Stochastic Systems", 3rd Conference on Robot Learning (CoRL), Osaka, Japan, 2019.

Published Journal papers:

- Yunpeng Pan, George I. Boutselis and Evangelos Theodorou,"Efficient Reinforcement Learning via Probabilistic Trajectory Optimization", on IEEE Transactions on Neural Networks and Learning Systems (TNNLS).

- George I. Boutselis, Yunpeng Pan and Evangelos Theodorou, "Numerical trajectory-optimization for stochastic mechanical systems", on SIAM Journal on Scientific Computing (SIAM-SISC).

Under review papers:

- George I. Boutselis and Evangelos Theodorou, "Differential Dynamic Programming on Lie groups: Derivation, Convergence analysis and Numerical results".

- George I. Boutselis, Marcus Pereira and Evangelos A. Theodorou, "Variational inference for stochastic control of infinite-dimensional systems".

- George I. Boutselis, Ziyi Wang and Evangelos A. Theodorou, "Constrained Sampling-based Trajectory Optimization using Stochastic Approximation".

Under preparation papers:

- George I. Boutselis, Yuichiro Aoyama and Evangelos Theodorou, “Constrained Differential Dynamic Programming with Lagrangian methods and optimality conditions”.

- George I. Boutselis, Yuichiro Aoyama and Evangelos Theodorou, “Data efficient model-based Reinforcement Learning by constrained Differential Dynamic Programming”.

- George I. Boutselis, Yuichiro Aoyama and Evangelos Theodorou, “Efficient chance-constrained trajectory optimization with constrained Differential Dynamic Programming”.

CHAPTER 2

TECHNICAL BACKGROUND

In this chapter we state some mathematical preliminaries and review background material including previous literature relating to our problems. Since we will be introducing concepts from various disciplines, each subsection will have its own notation.

2.1 Optimal control and trajectory optimization

We will present the general optimal control problem in its deterministic and stochastic settings. We will also give an overview of existing efficient, scalable algorithmic frameworks for solving such problems.

2.1.1 Discrete time, deterministic formulation

Let $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ be the state and control input vectors respectively, and denote $x(t_k) \equiv x^k$, $u(t_k) \equiv u^k$. Consider the performance index

$$J := \sum_{k=0}^{H-1} \mathcal{L}(x^k, u^k) + \mathcal{F}(x^H), \tag{2.1}$$

where t_0 is the starting time instant, t_H the final time, $\mathcal{F}(x)$ the terminal cost, and $\mathcal{L}(x, u)$ the running cost. The goal is to minimize J subject to the discrete-time dynamics $x^{k+1} = f(x^k, u^k), \quad x(t_0) = x^0$. Next, let us define the value function as follows

$$V(x^k, t^k) := \min_{u^k} J. \tag{2.2}$$

One of the cornerstones of optimal control theory is Bellman's principle of optimality. In discrete time this reads

$$V(x^k, t^k) = \min_{u^k} \big[\underbrace{\mathcal{L}(x^k, u^k) + V(x^{k+1}, t^{k+1})}_{Q(x^k, u^k)} \big], \tag{2.3}$$

Differential Dynamic Programming (DDP): DDP solves the optimal control problem numerically by using expansions of the dynamics and cost functions [3]. In this way, a complex nonlinear control problem can be simplified in the neighborhood of a nominal trajectory, and thus be efficiently solved in an iterative fashion. After propagating a nominal trajectory, the locally optimal controls and value function are computed in a backward pass. Based on these, a new state-control sequence is determined in a forward pass, and this trajectory is then treated as the new nominal trajectory for the next iteration. The aforementioned backward-forward passes are used iteratively until convergence to a (sub)optimal solution is reached. Due to the scalability and efficiency of the method, DDP-related algorithms have been widely used in robotics tasks [14].

Specifically, the derivation of DDP can be briefly described as follows: A Taylor expansion to the right-hand side of (2.3) will be used about nominal trajectories $\{\bar{x}^k, \bar{u}^k\}$, followed by a minimization with respect to control deviations δu, defined as: $u^k = \bar{u}^k + \delta u^k$. Then, we can find

$$\delta u_*^k = -{Q_{uu}}^{-1}(Q_{ux}\delta x^k + Q_u),$$

where the derivatives of Q are given by

$$Q_{xx} = \mathcal{L}_{xx}^k + (\Phi^k)^\top {V_{xx}}^{k+1}\Phi^k, \quad Q_{ux} = \mathcal{L}_{ux} + (B^k)^\top {V_{xx}}^{k+1}\Phi^k,$$

$$Q_{uu} = \mathcal{L}_{uu}^k + (B^k)^\top {V_{xx}}^{k+1}B^k, \quad Q_x = \mathcal{L}_x^k + (\Phi^k)^\top {V_x}^{k+1}, \quad Q_u = \mathcal{L}_u^k + (B^k)^\top {V_x}^{k+1}.$$

Φ and B above are the linearized state and control transition matrices of the dynamics.

The derivatives of the value function are found by plugging the above expressions into Bellman's principle of optimality, and quadratically expanding the value function. This will give

$$V_x{}^k = Q_x - Q_{ux}{}^\top Q_{uu}{}^{-1} Q_u, \quad V_{xx}{}^k = Q_{xx} - Q_{ux}{}^\top Q_{uu}{}^{-1} Q_{ux}.$$

2.1.2 Continuous time, stochastic formulation

We will briefly provide the corresponding definitions for the stochastic case in continuous time; that is, when the dynamics satisfy a stochastic differential equation. Keeping the same notation, the stochastic optimal control problem is

$$\min_u J(x,t), \quad \text{subject to} \quad \mathrm{d}x = (f(x) + G(x)u)\mathrm{d}t + B(x)\mathrm{d}w, \quad x(t_0) = x^0, \tag{2.4}$$

where w corresponds to a Brownian motion [15]. The cost and value function satisfy:

$$J(x,u) := \mathbb{E}\big[\int_{t_0}^{t_H} \mathcal{L}(x,u)\mathrm{d}t + \mathcal{F}(x(t_H))\big], \quad V(x,t) := \min_{u(t)} J(x,u), \tag{2.5}$$

$$V(x(t_k),t_k) = \min_{u(t_k)} \mathbb{E}\big[\int_{t_k}^{t_{k+1}} \mathcal{L}(x,u)\mathrm{d}t + V(x(t_{k+1}),t_{k+1})\big]. \tag{2.6}$$

Path Integral control (PI): When the cost is $\mathcal{L}(x,u) = q(x) + \frac{1}{2}u^\top Ru$, it can be shown that the optimal controls satisfy $u_* = -R^{-1}G^\top V_x(x,t)$. Under u_*, one can show that V satisfies the Hamilton-Jacobi-Bellman equation:

$$-V_t = q + V_x^\top f - \frac{1}{2}V_x^\top G R^{-1} G^\top V_x + \frac{1}{2}\text{trace}(V_{xx}BB^\top). \tag{2.7}$$

Therefore, finding u_* requires solving equation (2.7). This can rarely be done analytically and numerical solutions scale poorly when high-dimensional systems are considered. PI control avoids this issue by using an exponential transformation of the value function:

$\Psi(x,t) := \exp(-\frac{1}{\lambda}V(x,t))$, $\lambda > 0$. This transforms (2.7) into a linear PDE which can then be solved using the Feynman-Kac Lemma [15]. Specifically, the solution we obtain is in a form of an expectation over system trajectories [1]. This scalable and parallelizable approach has allowed PI control to be applied in challenging robotics problems [16].

Variational optimization-based control: A different but parallel approach for solving stochastic optimal control problems is through variational optimization frameworks. The starting point is the relation between *Free energy* and *Relative Entropy* in statistical physics. This relation has the following form:

$$\text{Free Energy} \leq \text{Work} - \text{Temperature} \times \text{Entropy} \tag{2.8}$$

In mathematical terms, denoting by $S(\cdot)$ the generalized entropy between two measures, (2.8) is written:

$$-\tfrac{1}{\rho}\log_e \mathbb{E}_{\mathcal{R}}\big[\exp(-\rho\mathcal{J})\big] \leq \big[\mathbb{E}_{\tilde{\mathcal{R}}}(\mathcal{J}) - \tfrac{1}{\rho}S(\tilde{\mathcal{R}}\big|\big|\mathcal{R})\big], \tag{2.9}$$

where $\mathbb{E}_{\mathcal{R}}, \mathbb{E}_{\tilde{\mathcal{R}}}$ denote expectations under probability measures $\mathcal{R}$, $\tilde{\mathcal{R}}$ respectively. It can be shown that the inequality above collapses to an equality when $\tilde{\mathcal{R}}$ is substituted for $\mathcal{R}^*$, which satisfies $\mathrm{d}\mathcal{R}^*(\omega) = \frac{\exp(-\rho\mathcal{J})\mathrm{d}\mathcal{R}(\omega)}{\int \exp(-\rho\mathcal{J})\mathrm{d}\mathcal{R}(\omega)}$ [17]. The connection to stochastic optimal control is made when (i) $\mathcal{J}$ is viewed as a state cost function, (ii) measures $\tilde{\mathcal{R}}$ and $\mathcal{R}$ are associated to paths generated by controlled and uncontrolled stochastic systems, and (iii) dynamics are affine in control. In this case, the free energy corresponds to the value function and the entropy term to a standard control cost. From an algorithmic point of view, we can parameterize our controls and try to approach the optimal path measure $\mathcal{R}^*$. Similar to PI, this yields sampling-based methods, which however avoid the computation of the desirability function and its derivative [17].

2.2 Basics of constrained optimization

Here, we will present certain fundamentals of constrained optimization which will be used in subsequent sections. A detailed exposition can be found in [10]. We will consider the general formulation of constrained optimization problems:

$$\min_{x\in\mathbb{R}^n} \quad f(x), \qquad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \leq 0, & i \in \mathcal{I}, \end{cases} \tag{2.10}$$

where $\mathcal{E}$, $\mathcal{I}$ are finite sets of indices corresponding to equality and inequality constraints respectively. Let us also define the *Lagrangian function* of (2.10) as

$$L(x, \lambda) = f(x) + \sum_{i\in\mathcal{E}\cup\mathcal{I}} \lambda_i c_i(x), \tag{2.11}$$

for some vector of real numbers, λ.

2.2.1 First-order optimality conditions

The necessary optimality conditions for the general constrained optimization problem are stated below.

Theorem 1 (Karush-Kuhn-Tucker (KKT) conditions)**.** *Suppose that x_* is a local solution to* (2.10)*, that the functions f and c_i are continuously differentiable and that the set of active constraints gradients is linearly independent. Then there is a Lagrange multiplier*

vector λ_* such that

$$\begin{aligned}
\nabla_x L(x_*, \lambda_*) &= 0 \\
c_i(x_*) &= 0, \quad \forall i \in \mathcal{E} \\
c_i(x_*) &\leq 0, \quad \forall i \in \mathcal{I} \\
\lambda_{i,*} &\geq 0, \quad \forall i \in \mathcal{I} \\
\lambda_{i,*} c_i(x_*) &= 0, \quad \forall i \in \mathcal{I} \cup \mathcal{E}.
\end{aligned}$$

2.2.2 Penalty methods

Some important methods for constrained optimization replace the original problem by a sequence of subproblems in which the constraints are represented by terms added to the objective. We will briefly present below two approaches of this type. Specifically, nonsmooth exact penalty methods will be presented, which are exact. This means that, for certain choices of their penalty parameters, a single minimization can yield the exact solution of the nonlinear programming problem. This property is desirable because it makes the performance of penalty methods less dependent on the strategy for updating the penalty parameter. Since the inherent nonsmoothness is difficult to handle within gradient-based methods, such an approach will be incorporated into the sampling-based methodologies developed in this thesis. Furthermore, the method of multipliers or augmented Lagrangian method will be presented, which will subsequently be used to extend DDP for constrained problems.

Nonsmooth penalty functions

Let us define the ℓ_1 *penalty function* corresponding to (2.10):

$$\phi(x; \mu) = f(x) + \mu \sum_{i \in \mathcal{E}} |c_i(x)| + \mu \sum_{i \in \mathcal{I}} [c_i(x)]^+, \tag{2.12}$$

where we use the notation $[y]^+ := \max(0, y)$. Then we have the following theorem:

Theorem 2. *Suppose that x_* is a strict local solution of the nonlinear programming problem* (2.10) *at which the first-order necessary conditions of Theorem 1 are satisfied, with Lagrange multipliers $\lambda_{*,i}$, $i \in \mathcal{E} \cup \mathcal{I}$. Then x_* is a local minimizer of $\phi(x; \mu)$ for all $\mu > \mu_*$ where*

$$\mu_* = ||\lambda_*||_\infty.$$

Loosely speaking, at a solution of the nonlinear program x_*, any move into the infeasible region is penalized sharply enough that it produces an increase in the penalty function to a value greater than $\phi(x_*; \mu) = f(x_*)$, thereby forcing the minimizer of $\phi(\cdot; \mu)$ to lie at x_*.

Augmented Lagrangian Method

Let us define the Augmented Lagrangian function corresponding to (2.10) as [18]

$$L_A(x, \lambda, \mu) := f(x) + \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \sum_{i \in \mathcal{E}} \frac{\mu_i}{2} c_i^2(x) + \sum_{i \in \mathcal{I}} \mathcal{P}(c_i(x), \lambda_{n_e+i}, \mu_{n_e+i}), \quad (2.13)$$

where λ, μ correspond to the Lagrange multipliers and penalty parameters respectively, while n_e denotes the total number of equality constraints. Moreover, the penalty function for inequalities, $\mathcal{P}$, is such that $\mathcal{P}'(y, \lambda, \mu) := \frac{\partial}{\partial y}\mathcal{P}(y, \lambda, \mu)$ is continuous for all $y \in \mathbb{R}$, $\lambda, \mu \in \mathbb{R}_{++}$ and: (i) $\mathcal{P}'(y, \lambda, \mu) \geq 0$, (ii) $\lim_{k\to\infty} \mu_{(k)} = \infty$ and $\lim_{k\to\infty} \lambda_{(k)} = \lambda > 0$ imply that $\lim_{k\to\infty} \mathcal{P}'(y_{(k)}, \lambda_{(k)}, \mu_{(k)}) = \infty$, (iii) $\lim_{k\to\infty} \mu_{(k)} = \infty$ and $\lim_{k\to\infty} \lambda_{(k)} = \lambda < 0$ imply that $\lim_{k\to\infty} \mathcal{P}'(y_{(k)}, \lambda_{(k)}, \mu_{(k)}) = 0$.

The algorithmic framework for the Augmented Lagrangian method is presented in Algorithm 1. Moreover, theorem 3 establishes the properties of this method, about which more details can be found in [18].

Algorithm 1 Augmented Lagrangian Method

1: Set $k \leftarrow 1$. Initialize $\epsilon_{(1)}$, $0 < \tau < 1$, $\gamma > 1$, $0 < \zeta < 1$, ω_*, ϵ_*.
2: Minimize $L_A(x, \lambda_{(k)}, \mu_{(k)})$, so that the approximate minimizer $x_{(k)}$ satisfies $||\nabla_x L_A(x_{(k)}, \lambda_{(k)}, \mu_{(k)})|| \leq \epsilon_{(k)}$
3: Update multipliers: $\lambda_{i,(k+1)} \leftarrow \mathcal{P}'(c_{i,(k)}, \lambda_{i,(k)}, \mu_{i,(k)})$, if $i \in \mathcal{I}$ and $\lambda_{i,(k+1)} \leftarrow \lambda_{i,(k)} + \mu_{i,(k)} c_{i,(k)}$ if $i \in \mathcal{E}$
4: Update inequality penalty parameters: For all $i \in \mathcal{I}$, if $\max[0, c_i(x_{(k+1)})] \leq \tau \max[0, c_i(x_{(k)})]$ and $|c_i(x_{(k+1)})\mu_{i,(k+1)}| \leq \tau |c_i(x_{(k)})\mu_{i,(k)}|$, then set $\mu_{i,(k+1)} \leftarrow \mu_{i,(k)}$, otherwise $\mu_{i,(k+1)} \leftarrow \gamma \mu_{i,(k)}$.
5: Update equality penalty parameters: For all $i \in \mathcal{E}$, if $|c_i(x_{(k+1)}| \leq \tau |c_i(x_{(k)})|$, then set $\mu_{i,(k+1)} \leftarrow \mu_{i,(k)}$, otherwise $\mu_{i,(k+1)} \leftarrow \gamma \mu_{i,(k)}$.
6: Set $\epsilon_{(k+1)} \leftarrow \zeta \epsilon_{(k)}$.
7: If $||\nabla_x L_A(x_{(k)}, \lambda_{(k)}, \mu_{(k)})|| \leq \epsilon_*$ and constraints are feasible within some tolerance ω_*, exit. Otherwise, set $k \leftarrow k + 1$ and go to Step 2.

Theorem 3. *Assume that $\{x_{(k)}\}$ is a sequence generated by Algorithm 1, and x_* is its limit point. Then one of the following possibilities hold:*

i) x_* is degenerate,

ii) x_* is a stationary (KKT) point of (2.10).

The most famous Augmented Lagrangian algorithm uses the penalty function

$$P(y, \lambda, \mu) = \frac{1}{2\mu}(\max(0, \lambda + \mu y)^2 - \lambda^2)$$

and is known as Powell-Hestenes-Rockafellar method. The main drawback of this approach is that the objective function of each subproblem is not twice differentiable, and thus may cause numerical instabilities when used, e.g., within Differential Dynamic Programming. Hence, we will investigate different penalty functions with continuous second order derivatives, such as the *hyperbolic barrier-quadratic* function

$$P(y, \lambda, \mu) = \frac{\lambda}{\mu}\theta(\mu y), \quad \theta(t) := \begin{cases} -\frac{1}{t}, & \text{if } t \leq -0.5 \\ 8t^2 + 12t + 6, & \text{otherwise.} \end{cases}$$

A set of possible penalty functions is given in [18].

2.3 Uncertainty quantification

Here we present methods which will allow us to represent uncertain systems. These will be used to quantify parametric uncertainty, as well as uncertainty due to lack of exploration. One of our directions of research will utilize the methodologies below to perform probabilistic control.

2.3.1 Polynomial Chaos & Karhunen-Loéve expansions

Let $\xi(\omega) \in \mathbb{R}^d$ be a continuous random variable with mutually independent components. We define the weighted L^2_ρ space as $L^2_\rho := \{\mathsf{f} : \mathcal{D} \to \mathbb{R} | \int_{\mathcal{D}} \mathsf{f}^2(\xi)\rho(\xi)\mathrm{d}\xi < \infty\}$, where $\mathcal{D}$ is the support of ξ, and $\rho(\xi)$ denotes the density function associated with P. The *Polynomial Chaos expansion* of a function $\mathsf{f} \in L^2_\rho$ is given by the series: $\mathsf{f}(\xi) = \sum_{j=0}^{\infty} \mathsf{f}_j \phi_j(\xi)$. Specifically, $\{\mathsf{f}_j,\ j \in \mathbb{Z}_{\geq 0}\}$ denote the Polynomial Chaos coefficients, while the polynomials $\{\phi_j(\xi),\ j \in \mathbb{Z}_{\geq 0}\}$ satisfy $\int_{\mathcal{D}} \phi_j(\xi)\phi_i(\xi)\rho(\xi)\mathrm{d}\xi = 0$, for each $i \neq j$.

In practice one has to truncate the expansion, as shown in (2.14). Therein, we also give an expression for computing the j^{th} Polynomial Chaos coefficient. This method relies on the orthogonality of $\{\phi_j\}$, and is called Galerkin projection:

$$\mathsf{f}(\xi) \approx \sum_{j=0}^{K} \mathsf{f}_j \phi_j(\xi), \quad \mathsf{f}_j = \frac{\int_{\mathcal{D}} \mathsf{f}(\xi)\phi_j(\xi)\rho(\xi)\mathrm{d}\xi}{\int_{\mathcal{D}} \phi_j^2(\xi)\rho(\xi)\mathrm{d}\xi}. \tag{2.14}$$

The series above can be viewed as an orthogonal projection of f onto the space spanned by $\{\phi_j,\ j = 0, ..., K\}$. Each ϕ_j is written as a product of univariate orthogonal polynomials. When these ϕ_j's are chosen to have total degree less than or equal to r, the number of coefficients will be given by: $K = \frac{(r+d)!}{r!d!} - 1$. As shown in [19], this constitutes the best polynomial approximation in the L^2_ρ norm.

Xiu and Karniadiakis developed the generalized Polynomial Chaos (gPC) framework in [5]. More precisely, they showed that when $\rho(\xi)$ is of a certain type, one can naturally select an appropriate orthogonal set, $\{\phi_j\}$, from the Askey scheme. Table 2.1 shows this

correspondence in detail. Each set $\{\phi_j\}$ forms a complete orthogonal basis in the Hilbert space determined by $\rho(\xi)$. One key element of gPC theory is that estimates for the moments of expanded objects can be computed analytically [5]. Unfortunately, the required gPC coefficients grow exponentially with the number of stochastic variables. Finally, we will often write for brevity $\langle \mathsf{f} \rangle := \mathbb{E}[\mathsf{f}] = \int_{\mathcal{D}} \mathsf{f}(\xi)\rho(\xi)\mathrm{d}\xi$, $\langle \mathsf{f}, \mathsf{g} \rangle := \mathbb{E}[\mathsf{fg}] = \int_{\mathcal{D}} \mathsf{f}(\xi)\mathsf{g}(\xi)\rho(\xi)\mathrm{d}\xi$ and so forth.

Table 2.1: Correspondence between distributions and orthogonal polynomials from the Askey scheme (α, β denote parameters of the corresponding density functions).

Distribution	Weight function	Polynomials	Domain
Gaussian	$e^{-\xi^2/2}$	Hermite	$(-\infty, \infty)$
Uniform	1	Legendre	$[-1, 1]$
Gamma	$\xi^{\alpha} e^{-\xi}$	Laguerre	$[0, \infty)$
Beta	$(1-\xi)^{\alpha}(1+\xi)^{\beta}$	Jacobi	$[-1, 1]$

While gPC expansions are used to approximate (functions of) random variables, the *Karhunen-Loéve* expansion can decompose continuous, bounded stochastic processes with finite variance:

$$Z(t,\omega) = \mu_Z(t) + \sum_{i=1}^{\infty} \sqrt{\zeta_i}\psi_i(t)\gamma_i(\omega), \quad \text{with} \quad \int_T C_Z(t,s)\psi_i(s)\mathrm{d}s = \zeta_i\psi_i(t), \quad (2.15)$$

$\forall t \in T$, $\forall i \geq 1$, and $\{\gamma_i(\omega)\}$ are mutually uncorrelated random variables with the following properties: $\mathbb{E}[\gamma_i] = 0$, $\mathbb{E}[\gamma_i\gamma_j] = \delta_{ij}$, $\gamma_i(\omega) = \frac{1}{\sqrt{\zeta_i}} \int_T \big(Z(t,\omega) - \mu_Z(t)\big)\psi_i(t)\mathrm{d}t$, $\forall i, j \geq 1$. The eigenvalue-eigenvector equation in (2.15) can be generally solved via numerical methods, while analytic solutions exist for particular types of covariance kernels. Furthermore, when the KL decomposition is utilized for Gaussian processes, $\{\gamma_i\}$ become independent Gaussian random variables [19].

As in gPC, one has to truncate the series in (2.15). The accuracy of the approximation will depend on the correlation length of the stochastic process. In fact, long correlation lengths result in fast decay of the eigenvalues ζ_i, meaning that only a few terms in (2.15) are required. This further implies that the KL transform can only be practically used for

colored random processes, since zero correlation length would require an infinite number of terms. Note that colored stochastic processes have found many applications in engineering (e.g., the Dryden wind turbulence model).

2.3.2 Gaussian processes

A *Gaussian process* (GP) is a collection of random variables, any finite subset of which has a joint Gaussian distribution [9]. Alternatively, a Gaussian process is defined as a distribution over function, or a generalization of the Gaussian distribution to an infinite-dimensional function space. Similar to a Gaussian distribution, a Gaussian process is completely specified by a mean function and a covariance function, i.e.,

$$p(f(x)) = \mathcal{GP}\Big(m(x), k(x, x')).$$

The covariance function $k(x, x')$ is also called a *kernel*. Without any prior knowledge of the model, we may assume a prior mean function $m(\cdot) = 0$ and a Squared Exponential (SE) covariance function plus a noise covariance $k(x_i, x_j) = \sigma_f^2 \exp(-\frac{1}{2}(x_i - x_j)^\top W(x_i - x_j)) + \sigma_\omega^2 \delta_{ij}$, where δ_{ij} is a Kronecker delta which is one iff $i = j$ and zero otherwise. $W = \text{diag}([\ l_1^2 \quad ... \quad l_{n+m}^2\ ])$. The hyperparameters θ of the kernel consist of the signal variance σ_f^2, the noise variance σ_ω^2 and the length scales for input space $l_1, ..., l_{n+m}$. The kernel function is interpreted as a similarity measure of random variables. In contrast to parametric approaches that rely on assumed structures and finite number of parameters, the GP approach puts a prior on function value directly, therefore it is nonparametric. GP inference exhibits significant practical limitations for learning and inference on large datasets due to its $O(N^3)$ computation and $O(N^2)$ space complexity, which is a direct consequence of having to store and invert a $N \times N$ matrix. This computational inefficiency is a bottleneck for applying GP-based RL in real-time.

Specifically, given a collection of sampled inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots \mathbf{x}_N\}$, and the corre-

sponding observations $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_N\}$, the joint distribution of the observed output and the output corresponding to a given test input $\mathbf{x}^*$ can be written as

$$\mathrm{p}\begin{pmatrix} \mathbf{F} \\ \mathbf{f}(\mathbf{x}^*) \end{pmatrix} = \mathcal{N}\Big(0, \begin{bmatrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_\omega^2 \mathbf{I} & \mathbf{k}(\mathbf{X},\mathbf{x}^*) \\ \mathbf{k}(\mathbf{x}^*,\mathbf{X}) & \mathbf{k}(\mathbf{x}^*,\mathbf{x}^*) \end{bmatrix}\Big),$$

where $\mathbf{K}$ is a matrix with entries $\mathbf{K}_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$. The posterior distribution, or predictive distribution, can be obtained by conditioning the joint distribution on the observed state transitions. Assuming independent outputs (no correlation between each output dimension), the predictive distribution is $\mathrm{p}(\mathbf{f}(\mathbf{x}^*)|\mathbf{x}^*, \mathbf{X}, \mathbf{F}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f)$ where the mean and variance are specified as

$$\begin{aligned} \boldsymbol{\mu}_f &= \mathbf{k}(\mathbf{x}^*,\mathbf{X})(\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_w \mathbf{I})^{-1}\mathbf{F}, \\ \boldsymbol{\Sigma}_f &= \mathbf{k}(\mathbf{x}^*,\mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*,\mathbf{X})(\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_w \mathbf{I})^{-1}\mathbf{k}(\mathbf{X},\mathbf{x}^*). \end{aligned} \tag{2.16}$$

2.4 Differential geometry and Lie groups

We review certain concepts from differential geometry and Lie group theory, which will be used for developing trajectory optimization methods on manifolds [20], [21] and [22].

We denote by G the Lie group that corresponds to the configuration space of a dynamical system. We let e be its identity element, and define $L_h : G \to G$ (respectively, $R_h : G \to G$) as the left (respectively, right) translation map, for all $h \in G$. The tangent and cotangent bundles of G are denoted by TG and T^*G, respectively, while $\mathfrak{g} := T_eG$ and $\mathfrak{g}^* := T_e^*G$ correspond to the Lie algebra and its dual. Given an n-dimensional Lie algebra, we define a linear isomorphism $\vee : \mathfrak{g} \to \mathbb{R}^n$, such that given an element $\xi \in \mathfrak{g}$ and a basis $\{E^i\}$ on $\mathfrak{g}$, we have $\xi^\vee = (\sum_{i=1}^n \xi^i E^i)^\vee := (\xi^1, ..., \xi^n)^\top$.The tangent map of L_h (resp., R_h) at $g \in G$ is written as $T_gL_h : TG \to TG$ (resp., $T_gR_h : TG \to TG$). We shall occasionally write for brevity hg and $g\xi$, ξg, instead of L_hg and $T_eL_g\xi$, $T_eR_g\xi$, for all $g, h \in G$, $\xi \in \mathfrak{g}$. Lastly, let $\mathfrak{X}$ represent the set of smooth vector fields on G. Then, for

any smooth function $\mathsf{f} : G \to \mathbb{R}$, we define the *Lie bracket* $[\cdot,\cdot] : \mathfrak{X} \times \mathfrak{X} \to \mathfrak{X}$, such that $[X,Y](\mathsf{f}) := X(Y(\mathsf{f})) - Y(X(\mathsf{f}))$. To proceed, we will also make use of the following notions:

Natural pairing and dual maps. Given a vector space V and its dual V^*, we define their *natural pairing* as the bilinear map $\langle\cdot,\cdot\rangle : V^* \times V \to \mathbb{R}$, such that $\langle \phi, x\rangle := \phi(x)$ for each $x \in V$, $\phi \in V^*$. Moreover, for any linear map $\mathsf{f} : V \to W$ between vector spaces, we define its *dual*, $\mathsf{f}^* : W^* \to V^*$, by imposing the property: $\langle \psi, \mathsf{f}(x)\rangle = \langle \mathsf{f}^* \circ \psi, x\rangle$, for each $\psi \in W^*$. Note that the former pairing is defined on (W^*, W), while the latter on (V^*, V). When $W = V^*$ and $\mathsf{f} = \mathsf{f}^*$, we say that f is a *symmetric* map. This implies the canonical identification of V with its bidual, V^{**}.

Affine connections. Let $X, Y \in \mathfrak{X}$ be two vector fields on a Lie group G. Given an affine connection, $\nabla : \mathfrak{X} \times \mathfrak{X} \to \mathfrak{X}$, we denote the covariant derivative of Y with respect to X by $\nabla_X Y$. A connection ∇ is termed *left-invariant* if it satisfies: $\nabla_{T_{(\cdot)}L_g X} T_{(\cdot)} L_g Y = T_{(\cdot)} L_g \nabla_X Y$, for all $g \in G$. We also define the *torsion* tensor of ∇, $\mathcal{T} : \mathfrak{X} \times \mathfrak{X} \to \mathfrak{X}$, as $\mathcal{T}(X,Y) := \nabla_X Y - \nabla_Y X - [X,Y]$. This term captures the difference between the Lie bracket and the utilized connection. When $\mathcal{T}(X,Y) = 0$ for all $X, Y \in \mathfrak{X}$, we say that ∇ is *symmetric*. For all left-invariant connections, there exists a bilinear map $\omega : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$, called the *connection function*, such that: $\nabla_{T_e L_g x} T_e L_g y = T_e L_g \omega(x,y)$, with $x, y \in \mathfrak{g}$. Of particular interest in this paper are the *Cartan-Schouten* connections. These are determined by $\omega(x,y) = \kappa[x,y]$, where $\kappa = 0$, $\kappa = 1$, and $\kappa = \frac{1}{2}$ correspond to the (-), (+), and (0) Cartan-Schouten connection, respectively. It can be shown that the (0) connection is a symmetric one [23].

Differential and Hessian operators. Consider a twice differentiable function $\mathsf{f} : G \to \mathbb{R}$, and let $\mathsf{x} \in T_g G$, $Y \in \mathfrak{X}$ so that $Y : G \to TG$. The *differential* of f at $g \in G$ is denoted by $\mathsf{Df}(g) : T_g G \to \mathbb{R}$ and satisfies: $\mathsf{x}(\mathsf{f}(g)) = \mathsf{Df}(g)(\mathsf{x})$. When necessary, we will use a suffix to indicate differentiation with respect to a specific argument (e.g., $\mathsf{D}_h \mathsf{f}$ is the differential of f with respect to h).

The *Hessian* operator, $\mathrm{Hessf}(g) : T_gG \to T_g^*G$, is a (0,2)-tensor which satisfies the identity: $\mathsf{D}(\mathsf{Df}(Y))(g)(\mathsf{x}) = \mathrm{Hessf}(g)(\mathsf{x})(Y(g)) + \mathsf{Df}(g)(\nabla_{\mathsf{x}}Y)$. In the literature, this mapping is also referred to as the second covariant derivative [22], or the geometric Hessian [23]. When a symmetric connection is used, the Hessian becomes symmetric at all points (i.e., $\mathrm{Hessf}(g) = (\mathrm{Hessf}(g))^*$, for all $g \in G$). We will often use a superscript to indicate the associated connection function (e.g., $\mathrm{Hess}^{(0)}\mathsf{f}(g)$ corresponds to the (0) Cartan connection).

Exponential map. The *exponential map*, $\exp : \mathfrak{g} \to G$, is a local diffeomorphism defined by: $\exp(\xi) := \gamma(1)$, with $\gamma : \mathbb{R} \to G$ satisfying $\dot{\gamma}(0) = \xi$. Its right-trivialized tangent, $\mathrm{dexp} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$, is determined so that: $\mathsf{D}\exp(\xi) \cdot \zeta = T_eR_{\exp(\xi)}\mathrm{dexp}_\xi\zeta$. We also define the *logarithm map*, $\log : G \to \mathfrak{g}$, as the inverse of $\exp(\cdot)$ (i.e., $\log(\exp(\xi)) = \xi$). When necessary, we will use a subscript to denote the group with respect to which the above operators are applied. For example, $\exp_G$ denotes the exponential map associated with G. This allows us to also differential between the group exponential map and the exponential function on $\mathbb{R}$.

In case of matrix Lie groups, one can think of G as a closed subset of $n \times n$ invertible matrices. The group operation will correspond to matrix multiplication, with e being the identity matrix. Moreover, the exponential map will be given by the standard matrix exponential, while dexp is determined as infinite series [24].

2.5 Discrete mechanics

Here, we present fundamental concepts from Lagrangian mechanics in continuous and discrete time. These will be subsequently combined with gPC theory to efficiently propagate systems under parametric uncertainties.

The continuous Lagrange-d'Alembert principle: Given a finite-dimensional system, let q, F and $L(q, \dot{q})$ denote its generalized position coordinates, non-conservative forces and

Lagrangian function, respectively. The Lagrange-d'Alembert principle imposes:

$$\delta \int_{t_0}^{t_f} L(q,\dot{q})\mathrm{d}t + \int_{t_0}^{t_f} F(q,\dot{q},u)\delta q \mathrm{d}t = 0.$$

From this, one can obtain the *Euler-Lagrange equations* of the system [7]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = F.$$

The continuous Pontryagin-d'Alembert principle: To proceed, let v, p denote the generalized velocity and momentum coordinates, respectively. The Pontryagin-d'Alembert principle connects the Lagrangian and Hamiltonian points of view, and enforces the condition: $\delta \int_{t_0}^{t_f} \big(L(q,v) + p\cdot(\dot{q}-v)\big)\mathrm{d}t + \int_{t_0}^{t_f} F(q,\dot{q},u)\delta q \mathrm{d}t = 0$. Manipulation of this law yields [6]:

$$v = \dot{q}, \quad p = \frac{\partial L}{\partial v}, \quad \dot{p} = \frac{\partial L}{\partial q} + F. \tag{2.17}$$

Equation (2.17) is equivalent to the Euler-Lagrange equations.

Discrete Lagrangian mechanics: Let q^k, u^k denote the discrete pose and control input at instant t_k, and let Δt be the (fixed) time-step. We define the *discrete Lagrangian*, L_d, such that $L_d(q^k, q^{k+1}) \simeq \int_{t_k}^{t_{k+1}} L(q(s),\dot{q}(s))\mathrm{d}s$. This quantity can be numerically determined via any quadrature method [25]. From a computational standpoint, one-point rules combine accuracy and efficiency. Specifically, we write: $L_d(q^k, q^{k+1}) = L((1-\zeta)q^k + \zeta q^{k+1}, \frac{q^{k+1}-q^k}{\Delta t})\Delta t$, such that $\zeta \in [0,1]$. When $\zeta \equiv 1/2$, the quadrature corresponds to the midpoint rule and is second-order accurate. A more involved expression can be used to increase the order of the resulting integrator [7]. In a similar manner, the continuous non-conservative forces are approximated by their left and right discrete counterparts as follows: $\int_{t_k}^{t_{k+1}} F(q(s),\dot{q}(s),u(s))\delta q \mathrm{d}s \simeq F_d^-(q^k, q^{k+1}, u^k)\delta q^k + F_d^+(q^k, q^{k+1}, u^k)\delta q^{k+1}$. One can typically choose [26]: $F_d^-(q^k, q^{k+1}, u^k) = F(\frac{q^k+q^{k+1}}{2}, \frac{q^{k+1}-q^k}{\Delta t}, u^k)\Delta t$, $F_d^+(q^k, q^{k+1}, u^k) = 0$. Plugging the above approximations into the Lagrange-d'Alembert principle, gives the

Discrete Euler-Lagrange equations (DEL) [7]:

$$0 = p^k + D_1 L_d(q^k, q^{k+1}) + F_d^-(q^k, q^{k+1}, u^k), \tag{2.18a}$$

$$p^{k+1} = D_2 L_d(q^k, q^{k+1}) + F_d^+(q^k, q^{k+1}, u^k). \tag{2.18b}$$

In this way, we obtain a Variational Integrator for propagating the discrete dynamics forward in time. Specifically, given (q^k, p^k), (2.18a) is solved implicitly for q^{k+1}, while (2.18b) determines p^{k+1} explicitly. Here, $D_i L_d(\cdot)$ denotes the partial derivative of L_d with respect to its i^{th} argument, and p^k can be viewed as the discrete momentum at $t \equiv t_k$.

It has been shown that variational integration methods generally outperform schemes that discretize the equations of motion directly [6, 7]. We will use such schemes to improve the performance of certain control algorithms derived in this proposal. Lastly, we note that VI's can also be developed for systems evolving on Lie groups as well - see [27] for details.

2.6 Stochastic partial differential equations

Let, H, U be separable Hilbert spaces and consider infinite-dimensional stochastic systems of the form:

$$\mathrm{d}X(t) = \mathcal{A}X(t)\mathrm{d}t + F\big(t, X(t)\big)\mathrm{d}t + \mathcal{G}\big(t, X(t)\big)\mathrm{d}W(t), \tag{2.19}$$

where $\mathcal{A} : D(\mathcal{A}) \subset H \to H$ is a linear operator ($D(\mathcal{A})$ denotes here the domain of $\mathcal{A}$). $F : H \to H$ and $G : U \to H$ are nonlinear operators that satisfy properly formulated Lipschitz conditions, associated with the existence and uniqueness of solutions for (2.19) (see [13, Theorem 7.2]). The term $W(t) \in U$ corresponds to a *Q-Wiener process*, which is a generalization of the Wiener process in infinite dimensions. When this noise profile is uncorrelated with respect to its spatial component, we will call it a *cylindrical Wiener process*. More details about these concepts are given in reference [13]. An im-

portant property that will be used in our algorithms, is the decomposition of $W(\cdot)$ into: $W(t) = \sum_{j=1}^{\infty} \sqrt{\lambda_j}\beta_j(t)e_j$, where $\{\beta_j(t)\}$ are mutually independent, real-valued Brownian motions, $\{\lambda_j\}$ are real numbers and $\{e_j\}$ form a complete orthonormal system in U. $\langle \cdot, \cdot \rangle_S$ denotes the inner product in a Hilbert space S (it will be clear when we refer to an inner product versus the natural pairing from section 2.4).

Furthermore, $C([0,T];H)$ will be the space of continuous processes in H for $t \in [0,T]$. We will use the notation $X(\cdot,\omega)$ to denote a state trajectory of the SPDE, and we will view the mapping $\omega \mapsto X(\cdot,\omega)$ as a $C([0,T];H)$-valued random variable [13, Section 3.7]. Many physical and engineering systems can be written in the abstract form of (2.19), by properly defining operators $\mathcal{A}$, F and $\mathcal{G}$ along with their corresponding domains; see[13, Chapter 13]).

Next, we state Girsanov's theorem for systems evolving in Hilbert spaces. We also include an essential part of its proof, since subsequent sections will rely on similar arguments to derive sampling-based controllers for stochastic partial differential equations (SPDEs). More details can be found in [13, Theorem 10.18].

Theorem 4 (Girsanov)**.** *Let Ω be a sample space with a σ-algebra $\mathcal{F}$. Consider the following H-valued stochastic processes:*

$$\mathrm{d}X = \big(\mathcal{A}X + F(t,X)\big)\mathrm{d}t + \mathcal{G}(t,X)\mathrm{d}W(t), \tag{2.20}$$

$$\mathrm{d}\tilde{X} = \big(\mathcal{A}\tilde{X} + F(t,\tilde{X})\big)\mathrm{d}t + \tilde{B}(t,\tilde{X})\mathrm{d}t + \mathcal{G}(t,\tilde{X})\mathrm{d}W(t), \tag{2.21}$$

where $X(0) = \tilde{X}(0) = x$ and $W \in U$ is a cylindrical Wiener process with respect to measure $\mathbb{P}$. Moreover, for each $\Gamma \in C([0,T];H)$, let the law of X be defined as $\mathcal{R}(\Gamma) := \mathbb{P}(\omega \in \Omega | X(\cdot,\omega) \in \Gamma)$. Similarly, the law of $\tilde{X}$ is defined as $\tilde{\mathcal{R}}(\Gamma) := \mathbb{P}(\omega \in \Omega | \tilde{X}(\cdot,\omega) \in \Gamma)$. Then

$$\tilde{\mathcal{R}}(\Gamma) = \mathbb{E}_{\mathbb{P}}\big[\exp\big(\int_0^T \langle \psi(s), dW(s)\rangle_U - \tfrac{1}{2}\int_0^T ||\psi(s)||_U^2 ds\big) | X(\cdot) \in \Gamma\big], \tag{2.22}$$

where we have defined $\psi(t) := \mathcal{G}^{-1}\big(t, X(t)\big)\tilde{B}\big(t, X(t)\big) \in U_0$*. We have also assumed that* $\mathbb{E}_{\mathbb{P}}\big[e^{\frac{1}{2}\int_0^T ||\psi(t)||^2 \mathrm{d}t}\big] < +\infty$*.*

Proof 1. *Define the process:*

$$\hat{W}(t) := W(t) - \int_0^t \psi(s)\mathrm{d}s. \tag{2.23}$$

Under the aforementioned assumptions, $\hat{W}$ *is a cylindrical Wiener process with respect to a measure* $\mathbb{Q}$ *determined by:*

$$\mathrm{d}\mathbb{Q}(\omega) = \exp\big(\int_0^T \langle\psi(s), \mathrm{d}W(s)\rangle_U - \frac{1}{2}\int_0^T ||\psi(s)||_U^2 \mathrm{d}s\big)\mathrm{d}\mathbb{P} \tag{2.24}$$

The proof for this result can be found in [13, Theorem 10.14]. Now, using (2.23)*,* (2.20) *will be rewritten as:*

$$\mathrm{d}X = \big(\mathcal{A}X + F(t,X)\big)\mathrm{d}t + \mathcal{G}(t,X)\mathrm{d}W(t) \tag{2.25}$$

$$= \big(\mathcal{A}X + F(t,X)\big)\mathrm{d}t + B(t,X)\mathrm{d}t + \mathcal{G}(t,X)\mathrm{d}\hat{W}(t) \tag{2.26}$$

Notice that the SPDE in (2.26) *has the same form as* (2.21)*. Therefore, under the introduced measure* $\mathbb{Q}$ *and noise profile* $\hat{W}$*,* $X(\cdot,\omega)$ *becomes equivalent to* $\tilde{X}(\cdot,\omega)$ *from* (2.21)*. Conversely, under measure* $\mathbb{P}$*,* (2.25) *(or* (2.26)*) behaves as the original system in* (2.20)*. In other words,* (2.20) *and* (2.26) *describe the same system on* $(\Omega, \mathcal{F}, \mathbb{P})$*. From the uniqueness of solutions and the aforementioned reasoning, one has:*

$$\mathbb{P}\big(\{\tilde{X} \in \Gamma\}\big) = \mathbb{Q}\big(\{X \in \Gamma\}\big).$$

The result follows from (2.24)*.* ■

2.7 Discrete Control Barrier Functions (CBFs)

Finally, we briefly describe Control Barrier Functions for safe control and tracking of discrete-time systems [28]. Let the control-affine system

$$x^{k+1} = F(x^k) + G(x^k)u^k, \tag{2.27}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and the safe set is defined, given a function $B :\in \mathbb{R}^n \to \mathbb{R}$, as

$$\mathcal{D} := \{x \in \mathbb{R}^n | B(x) \geq 0\}. \tag{2.28}$$

The (exponential) control barrier function is such that:

$$\begin{aligned} &B_{k+1} + (\gamma - 1)B_k \geq 0 \\ &B_0 \geq 0, \end{aligned} \tag{2.29}$$

given $0 \leq \gamma \leq 1$. If eq. (2.29) is satisfied for all t_k, it is easy to show that $\mathcal{D}$ is forward invariant. When $B(x)$ is either linear or quadratic in x, constraint (2.29) can be formulated as linear or quadratic in u via the control-affine dynamics in (2.27). Hence, when $B(x) = x^\top Px + H^\top x$, for $P \succeq 0$, we can formulate controllers based on (2.29) for safe tracking as

$$\begin{aligned} \min_{u,d} \quad & (||u^k||^2 + Kd) \\ \text{s.t.} \quad & CLF + d \leq 0 \\ & (u^k)^\top Q u^k + r^\top u^k + s \leq 0 \\ & u_{min} \leq u^k \leq u_{max} \\ & d \geq 0, \end{aligned} \tag{2.30}$$

where CLF corresponds to a Control Lyapunov condition (e.g., for tracking a desired trajectory), which for a quadratic Lyapunov function, is again quadratic in u^k. Moreover, Q, r, s above will depend on P, H, the transition matrices in (2.27), as well as the current state x^k. Finally, d denotes a relaxation parameter to the tracking objective, in case the CLF and CBF conditions contradict with each other, which is penalized via a large positive constant K. Problem (2.30) is a convex, quadratically-constrained quadratic program (QCQP) which can be efficiently solved with interior point methods, and thus allow for real-time safe control.

CHAPTER 3

LINEARIZATION-BASED TRAJECTORY OPTIMIZATION AND PROBABILISTIC CONTROL

We have given in section 2.1.1 the basics for deriving the unconstrained version of Differential Dynamic Programming (DDP). This algorithm will be extended in this chapter to incorporate uncertainty quantification methods, as well as handle systems evolving on Lie groups. We will begin, however, by discussing a set of constrained DDP methods for deterministic systems.

3.1 Constrained Differential Dynamic Programming

3.1.1 Methodology I: Using KKT conditions

Let us consider the generic dynamic system

$$x^k = \begin{bmatrix} p^k \\ v^k \end{bmatrix} \in \mathbb{R}^n, \quad x^{k+1} = \begin{bmatrix} f_p(x^k) \\ f_v(x^k, u^k) \end{bmatrix}, \tag{3.1}$$

where $u^k \in \mathbb{R}^m$ are the controls and $p^k \in \mathbb{R}^{n_p}$, $v^k \in \mathbb{R}^{n_v}$ correspond to "position" and "velocity" states respectively. Notice that the controls affect directly only the transition of the velocity dynamics. Most mechanical/robotics systems can be described by such equations of motion in discrete time.

Our goal is to minimize the following problem:

$$\begin{aligned} \min_u J &= \min_u \big[\sum_{k=0}^{H-1} \mathcal{L}(x^k, u^k) + \mathcal{F}(x^H) \big] \\ \text{subject to:} \quad & \text{dynamics constraints (3.1)} \\ & g_p^k(p^k) \leq 0 \text{ and } g_v^k(v^k) \leq 0, \forall k. \end{aligned} \tag{3.2}$$

$g_p^k(\cdot)$ and $g_v^k(\cdot)$ above correspond to inequality constraints of position and velocity states respectively. We can also account for equality constraints, but will omit them for brevity.

We will start by using Bellman's principle of optimality for the constrained case at t_k to express the value function V (see 2.1.1 for definitions):

$$\begin{aligned} V(x^k, t^k) &= \min_{u^k} \big[\underbrace{\mathcal{L}(x^k, u^k) + V(x^{k+1}, t^{k+1})}_{Q(x^k, u^k)} \big] \\ \text{s.t.} \quad & g_v^{k+1}(v^{k+1}) \leq 0, \quad g_p^{k+2}(p^{k+2}) \leq 0. \end{aligned} \tag{3.3}$$

Notice that we are using the one-step-ahead and two-step-ahead constraints for the velocities and positions respectively. This is because these constraints are the ones directly affected by control u^k. To see this, notice that

$$v^{k+1} = f_v(x^k, u^k) \quad \text{and} \quad p^{k+2} = f_p(f_p(x^k), f_u(x^k, u^k)).$$

We proceed with the backward pass of DDP. Let the active constraints along the nominal trajectory be $\hat{\boldsymbol{g}}$. We assume that these constraints will remain active under small perturbations. Linearizing about the nominal trajectory $\{\bar{x}^k, \bar{u}^k\}$ we have

$$\hat{\boldsymbol{g}}(\bar{x}^k + \delta x^k, \bar{u}^k + \delta u^k) \approx \hat{\boldsymbol{g}}(\bar{x}^k, \bar{u}^k) + \underbrace{\nabla_u \hat{\boldsymbol{g}}(\bar{x}^k, \bar{u}^k)}_{\boldsymbol{C}^k} \delta u^k + \underbrace{\nabla_x \hat{\boldsymbol{g}}(\bar{x}^k, \bar{u}^k)}_{-\boldsymbol{D}^k} \delta x^k \tag{3.4}$$

Based on the assumption that these constraint will remain active, we enforce

$$\boldsymbol{C}^k \delta u^k = \boldsymbol{D}^k \delta x^k. \tag{3.5}$$

Now by approximating the value function about the nominal trajectory as in section 2.1.1 (see definitions of derivatives of Q therein), we consider the approximate version of (3.3):

$$\begin{aligned} \min_{\delta \boldsymbol{u}} [\frac{1}{2} \delta u^{\mathrm{T}} \boldsymbol{Q}_{\boldsymbol{uu}} \delta u + \delta u^{\mathrm{T}} \boldsymbol{Q}_{\boldsymbol{ux}} \delta x + \boldsymbol{Q}_{\boldsymbol{u}}^{\mathrm{T}} \delta u] \\ \text{subject to} \qquad \boldsymbol{C} \delta u = \boldsymbol{D} \delta \boldsymbol{x} \end{aligned} \tag{3.6}$$

where we have dropped the time indices for brevity. To solve this problem, construct the corresponding Lagrangian function as

$$L = \frac{1}{2} \delta \boldsymbol{u}^{\mathrm{T}} \boldsymbol{Q}_{\boldsymbol{uu}} \delta \boldsymbol{u} + \delta \boldsymbol{u}^{\mathrm{T}} \boldsymbol{Q}_{\boldsymbol{ux}} \delta \boldsymbol{x} + \boldsymbol{Q}_{\boldsymbol{u}} \delta \boldsymbol{u} + \boldsymbol{\lambda}^{\mathrm{T}} (\boldsymbol{C} \delta \boldsymbol{u} - \boldsymbol{D} \delta \boldsymbol{x}) \tag{3.7}$$

Where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. By the KKT conditions, we obtain

$$\begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{uu}} & \boldsymbol{C}^{\mathrm{T}} \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \delta \boldsymbol{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{Q}_{\boldsymbol{ux}} \\ \boldsymbol{D} \end{bmatrix} \delta \boldsymbol{x} - \begin{bmatrix} \boldsymbol{Q}_{\boldsymbol{u}} \\ \boldsymbol{0} \end{bmatrix} \tag{3.8}$$

Solving this equation via the Schur complement, we obtain

$$\delta \boldsymbol{u} = -\boldsymbol{H} \boldsymbol{Q}_{\boldsymbol{u}} + (-\boldsymbol{H} \boldsymbol{Q}_{\boldsymbol{ux}} + \boldsymbol{W}^{\mathrm{T}} \boldsymbol{D}) \delta \boldsymbol{x} \tag{3.9}$$

$$\boldsymbol{W} := (\boldsymbol{C} \boldsymbol{Q}_{\boldsymbol{uu}} \boldsymbol{C}^{\mathrm{T}})^{-1} \boldsymbol{C} \boldsymbol{Q}_{\boldsymbol{uu}}^{-1} \tag{3.10}$$

$$\boldsymbol{H} := \boldsymbol{Q}_{\boldsymbol{uu}}^{-1} (\boldsymbol{I} - \boldsymbol{C}^{\mathrm{T}} \boldsymbol{W}) \tag{3.11}$$

Before we proceed to the forward pass of constrained DDP, note that the linearizations

of the constraints can be computed via the chain rule. Specifically

$$\frac{\partial g_p^{k+2}}{\partial x^k} = \frac{\partial g_p^{k+2}}{\partial p^{k+2}} \frac{\partial p^{k+2}}{\partial x^{k+1}} \frac{\partial x^{k+1}}{\partial x^k} \tag{3.12}$$

$$\frac{\partial g_p^{k+2}}{\partial u^k} = \frac{\partial g_p^{k+2}}{\partial p^{k+2}} \frac{\partial p^{k+2}}{\partial x^{k+1}} \frac{\partial x^{k+1}}{\partial u^k} \tag{3.13}$$

and similarly for the velocity constraints.

Now for the forward pass, since the above derivation relies on linear/quadratic approximations of the cost and constraints, we will be solving the following QP problem at each t_k

$$\begin{aligned} &\underset{\delta u^k}{\operatorname{argmin}} [\frac{1}{2} {\delta u^k}^{\mathrm{T}} \boldsymbol{Q_{uu}} \delta u^k + \boldsymbol{Q_u}^{\mathrm{T}} \delta u^k + {\delta u^k}^{\mathrm{T}} \boldsymbol{Q_{ux}} \delta x^k] \\ &\text{s.t.} \quad \boldsymbol{g}(\bar{x}^k + \delta x^k, \bar{u}^k + \delta u^k) \approx \boldsymbol{g}(\bar{x}^k, \bar{u}^k) + \frac{\partial \boldsymbol{g}}{\partial x^k} \delta x^k + \frac{\partial \boldsymbol{g}}{\partial u^k} \delta u^k \leq 0 \end{aligned} \tag{3.14}$$

where we only consider constraints directly affected by u^k, whose derivatives can be computed as shown above via the chain rule. We note that the QP above may lead to infeasible trajectories, even though the linearized constraints are satisfied. In case this happens, we add a trust region on δu^k which we decrease overt the iterations. Hence, when we start with a feasible trajectory, this approach will result into an optimized trajectory that satisfies the constraints.

Simulations

We provide examples of the following systems: point mass in two dimensions, vehicle in two dimensions and quadrotor. We will compare our approach ("Modified") against the KKT-based algorithm in [29] ("Former"). The main difference of the previous approach with ours is that we consider both one-step ahead and current constraints in the forward and backward pass of DDP.

-*Point mass:* The goal is to find an optimized acceleration sequence $\boldsymbol{u} = [a_x, a_y]^{\mathrm{T}}$. The start

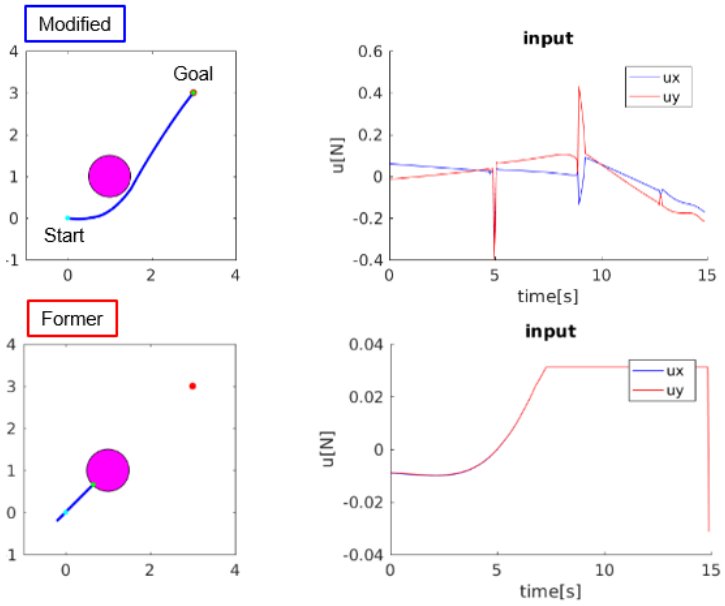


Figure 3.1: Results of KKT-based DDP on point mass.

point is $\boldsymbol{x}_{\mathrm{s}}$=$[0, 0, 0, 0]^{\mathrm{T}}$ and the target point is $\boldsymbol{x}_{\mathrm{g}}$=$[3, 3, 0, 0]^{\mathrm{T}}$. The initial trajectory will be a linear rollout from $\boldsymbol{x}_{\mathrm{s}}$ to $[0, 0.01]^{\mathrm{T}}$, and the initial control trajectory will be $\boldsymbol{u}_0 = [\mathbf{0}, \cdots, \mathbf{0}]$. The dynamics equations are

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ v^x_{k+1} \\ v^y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + v^x_k \Delta t \\ y_k + v^y_k \Delta t \\ v^x_k + a^y_k \Delta t \\ v^y_k + a^y_k \Delta t \end{bmatrix} \tag{3.15}$$

The result is shown in Fig. 3.1. Using our new method, the point mass could reach the goal.

Simple vehicle system: We will optimize over the steering angle and acceleration control sequence $\boldsymbol{u} = [u^\theta, u^v]^{\mathrm{T}}$. The start and target points are respectively $\boldsymbol{x}_{\mathrm{s}} = [0, 0, 0, 0]^{\mathrm{T}}$ to

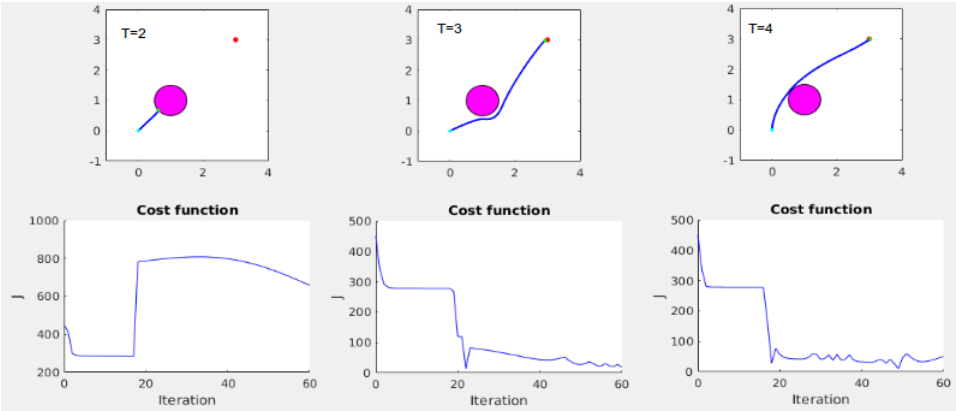


Figure 3.2: time horizon and trajectory

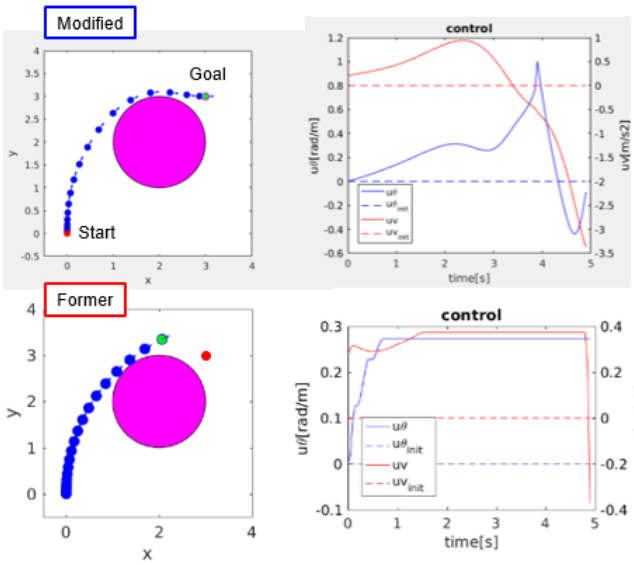


Figure 3.3: Results of KKT-based DDP on simplified vehicle model.

$\boldsymbol{x}_{\text{goal}}$=$[3, 3, \frac{\pi}{2}, 0]^{\mathrm{T}}$. Its dynamics is

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + v_k \sin\theta_k \Delta t \\ y_k + v_k \cos\theta_k \Delta t \\ \theta_k + u_k^{\theta} v_k \Delta t \\ v_k + u^{v} \Delta t \end{bmatrix} \tag{3.16}$$

The initial control and state trajectory were $\boldsymbol{u}_0 = [\mathbf{0}, \cdots, \mathbf{0}]$ and $\boldsymbol{x}_0 = [\mathbf{0}, \cdots, \mathbf{0}]$. The results are shown in Fig.3.3. Our approach clearly outperforms the previous work.

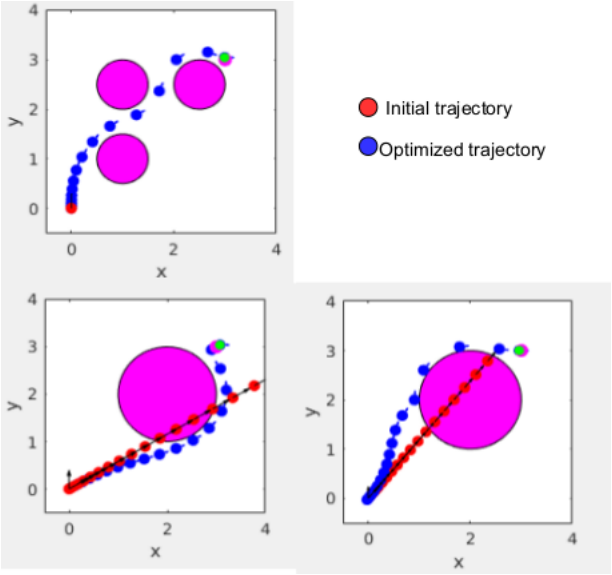


Figure 3.4: Results of KKT-based DDP on point mass.: Multiple obstacles and bad initial trajectory.

Quadrotor: The goal is to find an optimal control sequence to move a quadrotor from an initial trajectory that corresponds to hovering at $\boldsymbol{x_0} = [-3.5, 0, 0]^{\mathrm{T}}$ to the target point $\boldsymbol{x}_{\mathrm{goal}} = [2.8, 0, 0]^{\mathrm{T}}$, while avoiding voiding a static sphere obstacle. The dynamics of the system are

$$\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{v}_{k+1} \\ \boldsymbol{\theta}_{k+1} \\ \boldsymbol{\omega}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k + \boldsymbol{v}_k dt \\ \boldsymbol{v}_k + (\boldsymbol{g} + \frac{1}{m}\boldsymbol{R}\boldsymbol{f} - k_d \boldsymbol{v}_k)dt \\ \boldsymbol{\theta}_k + \boldsymbol{J}\omega_k dt \\ \omega_k + \boldsymbol{I}^{-1}\boldsymbol{\tau} dt \end{bmatrix} \tag{3.17}$$

We used lift force generated by each rotor as control $\boldsymbol{u} = [u_1, u_2, u_3, u_4]^{\mathrm{T}}$. Force $\boldsymbol{f}$ and torque $\boldsymbol{\tau}$ are calculated as

$$\boldsymbol{f} = \begin{bmatrix} 0 \\ 0 \\ u_1 + u_2 + u_3 + u_4 \end{bmatrix}, \qquad \boldsymbol{\tau} = \begin{bmatrix} l(u_3 - u_1) \\ l(u_2 - u_4) \\ \frac{b}{k}(u_1 - u_2 + u_3 - u_4) \end{bmatrix} \tag{3.18}$$

where:

m : mass of quadrotor

k_d : air resistance constant

l : length between cog of quadrotor and rotor

k : lift constant

b : drag constant

$\boldsymbol{I}$: inertia matrix of quadrotor

$\boldsymbol{R}$ and $\boldsymbol{J}$ are

$$\boldsymbol{R} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{3.19}$$

$$\boldsymbol{J} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \tag{3.20}$$

The result is shown in Fig. 3.5. It was observed that a good initial trajectory was required for the former algorithm, while our method could solve the problem only by starting from a hovering trajectory at $[0, 0, 0]^{\mathrm{T}}$.

3.1.2 Methodology II: Using Multiplier methods

Here, we will be using the Augmented Lagrangian approach to extend DDP for solving (3.2). The main idea is to observe that *partial elimination of constraints* can be used on the inequality constraints g of (3.2). This means that the penalty function $\mathcal{P}$ from section 2.2 can only applied to the inequality state constraints, while leaving the dynamics as hard

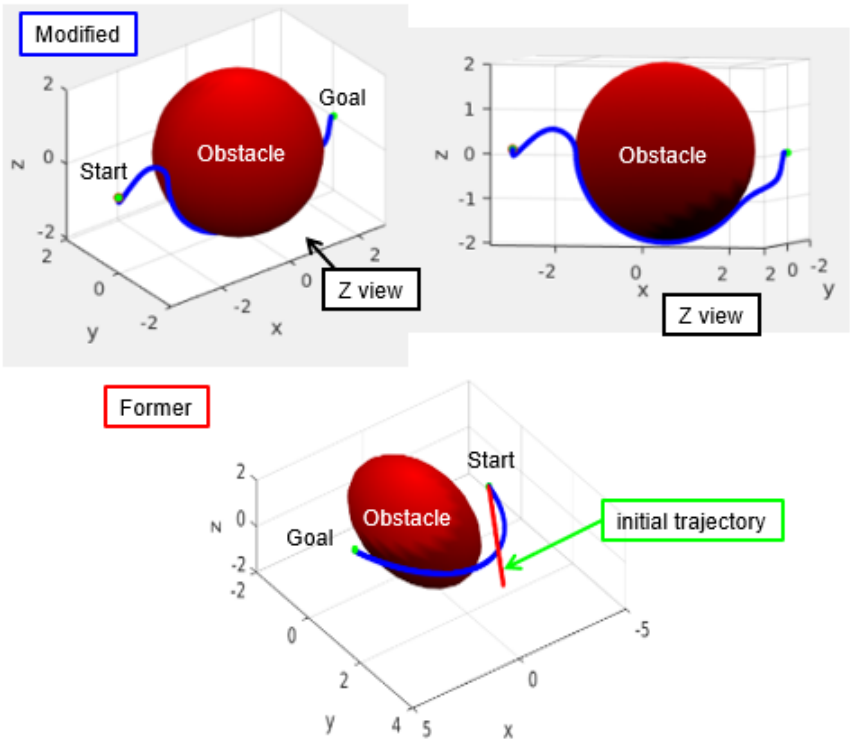


Figure 3.5: Quadrotor results.

constraints for each subproblemh.

We will thus be considering the following problem

$$\min_{U} \underbrace{\left[J(X,U) + \sum_{i,k} \mathcal{P}(\lambda_i^k, \mu_i^k, g_i^k(x^k)) \right]}_{L_A} \tag{3.21}$$

$$\text{s.t.} \quad x^{k+1} = f(x^k, u^k), \quad x^0 = \bar{x}^0, \quad k = 0, 1, ..., H-1,$$

where λ_i^k, μ_i^k denote Lagrange multipliers and penalty parameters respectively, X, U denote here the stacked states and controls, while $\mathcal{P}(\cdot)$ denote properly defined penalty functions. We will thus be using the following algorithm:

Since DDP requires $L_A(\cdot)$ to be twice differentiable, we selected the penalty function as

$$\mathcal{P}(\lambda_i^k, \mu_i^k, g_i^k(x^k)) = \frac{(\lambda_i^k)^2}{\mu_i^k} \phi\left(\frac{\mu_i^k}{\lambda_i^k} g_i^k(x^k) \right),$$

Algorithm 2 Augmented Lagrangian DDP

1: Set $i \leftarrow 1$. Initialize $\epsilon_{(1)}$, $0 < \tau < 1$, $\gamma > 1$, $0 < \zeta < 1$, ω_*, ϵ_*.
2: Minimize (3.21) using unconstrained DDP from section 2.1.1, so that the approximate minimizer $U_{(i)}$ satisfies $||\nabla_U L_A(\{u^k_{(i)}, \lambda^k_{(i)}, \mu^k_{(i)}\})|| \leq \epsilon_{(i)}$
3: Update multipliers: $\lambda^k_{j,(i+1)} \leftarrow \mathcal{P}'(g^k_{j,(i)}, \lambda^k_{j,(i)}, \mu^k_{j,(i)})$
4: Update inequality penalty parameters: If $\max[0, g^k_j(U_{(i+1)})] \leq \tau \max[0, g^k_j(U_{(i)})]$ and $|g^k_j(U_{(i+1)})\mu^k_{k,(i+1)}| \leq \tau |g^k_j(U_{(i)})\mu^k_{j,(i)}|$, then set $\mu^k_{j,(i+1)} \leftarrow \mu^k_{j,(i)}$, otherwise $\mu^k_{j,(j+1)} \leftarrow \gamma\mu^k_{j,(i)}$.
5: Set $\epsilon_{(i+1)} \leftarrow \zeta\epsilon_{(i)}$.
6: If $||\nabla_U L_A(U_{(i)}, \lambda_{(i)}, \mu_{(i)})|| \leq \epsilon_*$ and constraints are feasible within some tolerance ω_*, exit. Otherwise, set $i \leftarrow i + 1$ and go to Step 2.

with

$$\phi(t) := \begin{cases} \frac{1}{2}t^2 + t, & t \geq -\frac{1}{2} \\ -\frac{1}{4}\log(-2t) - \frac{3}{8}, & \text{otherwise,} \end{cases}$$

which can be viewed as a smooth approximation to the Powell-Hestenes-Rockafellar method.

Simulations

We present the results of the Augmented Lagrangian DDP on the following systems: Inverted pendulum, Cartpole and Quadrotor with state constraints. The algorithm was terminated when $Q_u \leq 1e-3$ and $||g^k_i|| \leq 1e-4$ for all i, k. The state constraints considered are given as bounds (depicted by magenta lines in figures) for the first two systems, and four spherical objects for the last one.

3.1.3 Combining previous methodologies for improved convergence of constrained DDP

It can be observed from the simulated results that the above approaches share different benefits and flaws. Specifically, the performance of the KKT-based methodology can be highly affected by a bad initialization, which may require many iterations of the algorithm to converge. This can be viewed in figure 3.2. An intuitive explanation might be that each subproblem (3.3) only considers the current control, u^k, and not the entire trajectory, which may lead to bad overall performance. However, as iterations progress, Bellman's principle

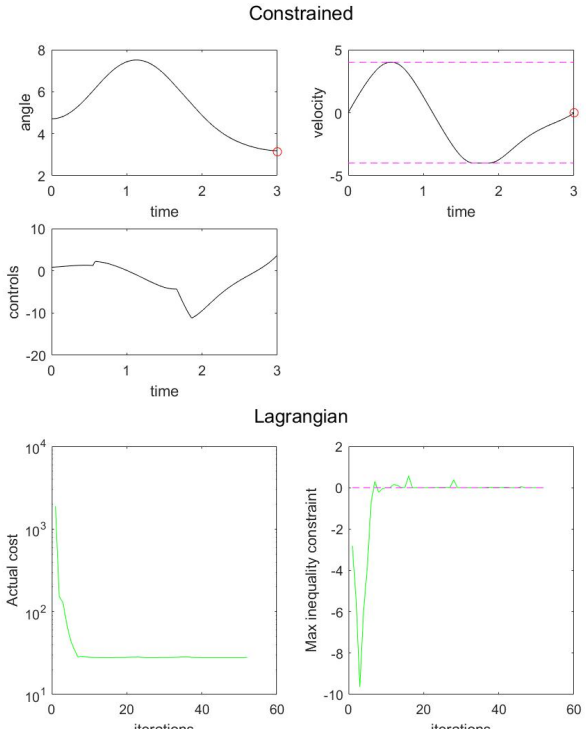


Figure 3.6: Augmented Lagrangian: Inverted pendulum

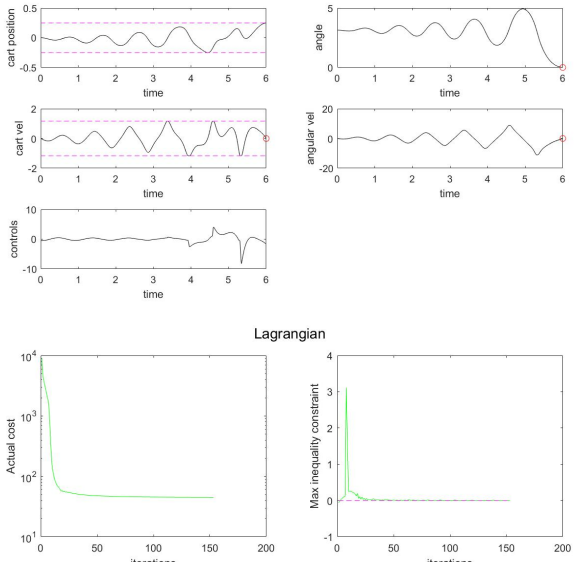


Figure 3.7: Augmented Lagrangian: Cartpole

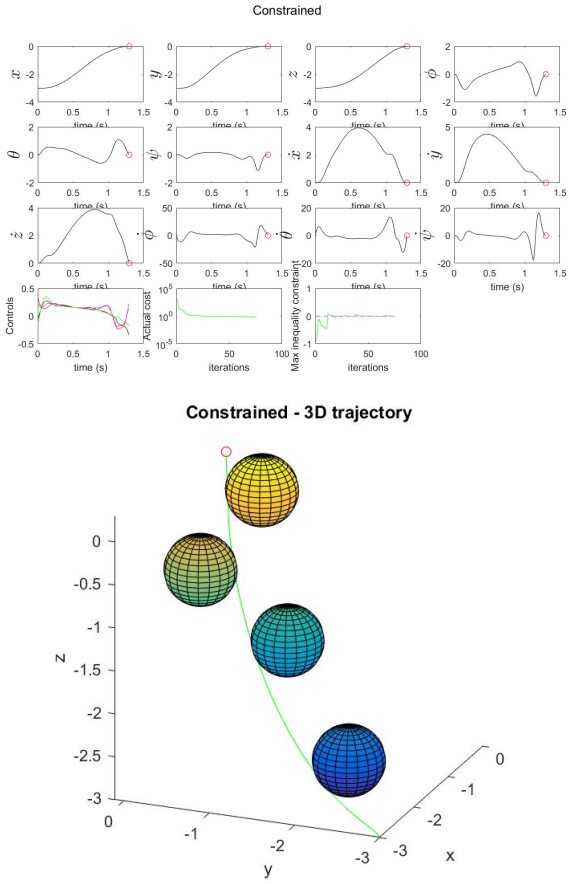


Figure 3.8: Augmented Lagrangian: Quadrotor

Table 3.1: Comparison of constrained trajectory optimization methods: Elapsed timed in seconds for simple vehicle model - horizon of 100 steps.

Augmented Lagrangian DDP	KKT-based DDP	AL-DDP + KKT-DDP	SQP
5.85	5.3	**1.7**	6.07

Table 3.2: Comparison of constrained trajectory optimization methods: Elapsed timed in seconds for cart pole - horizon of 300 steps.

Augmented Lagrangian DDP	KKT-based DDP	AL-DDP + KKT-DDP	SQP
18	8	**3.58**	>60

allows for fast convergence since the initial optimization problem is splitted into multiple subproblems.

In contrast, the Augmented Lagrangian approach is pretty robust to initialization, but displays oscillatory behavior near the (local) solution. This can be readily explained from optimization theory, since Multiplier methods generally converge only linearly. This behavior is depicted in figures 3.7, 3.6, 3.8.

The idea here is to combine the two approaches: We begin by using algorithm 2 until a pre-specified precision of the cost and constraints, and subsequently switch to the KKT-based approach of section 3.1.1. If sufficient improvement is not observed within a few iterations, we switch back to the Augmented Lagrangian method, and reduce the aforementioned tolerances for the "switching" mechanism.

Simulations

We compare the two individual versions of constrained DDP we presented against the switching mechanism discussed here on two systems. We give the elapsed time required for each algorithm to achieve the prespecified tolerances. All values correspond to MATLAB implementations of the methods. We also compare against the built-in MATLAB SQP algorithm. The results can be found in tables 3.2 and 3.1.

3.2 Control of systems with parametric uncertainties

In this line of research we will control dynamic systems influenced by uncertain internal parameters and initial states through gPC theory. gPC has been shown to be more efficient than Monte Carlo-based alternatives and thus has found applications in many fields [19]. However, works on control-related problems are rather limited. Hover and Triantafyllou utilized gPC as a tool to analyze the stability of a stochastic bilinear system [30]. Moreover, Fisher and Bhattacharya in [31] proposed a stochastic version of the LQR controller. In contrast, we will develop control methods of nonlinear stochastic systems [32, 33, 34].

3.2.1 Unconstrained control

We consider optimal control problems of the form

$$\begin{aligned} &\min_{u} \int_{t_0}^{t_f} \mathcal{L}(\mathcal{M}(t), u, t)\mathrm{d}t + \mathcal{F}(\mathcal{M}(t_f), t_f) \\ &\text{s.t.} \quad \dot{x}(t) = f(x(t), u(t), t; \lambda^p), \quad x(t_0) = \bar{x}^0(\lambda^0). \end{aligned} \tag{3.22}$$

Here, $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, and $\mathcal{L}$, $\mathcal{F}$ denote the running and terminal cost. In addition, $\lambda^p \in \mathbb{R}^{d_p}$, $\lambda^0 \in \mathbb{R}^{d_0}$ are independent random variables associated with uncertainty in model parameters and initial states, respectively. We assume that the statistics of these latter variables are known a priori. Finally, $\mathcal{M} := (\mathcal{M}_1, ..., \mathcal{M}_n)^\top$ contains the statistical moments of our state, with $(\mathcal{M}_i)_j$ being the j^{th} (central) moment of x_i, and $\mathcal{M}_i := ((\mathcal{M}_i)_1, ..., (\mathcal{M}_i)_j, ..., (\mathcal{M}_i)_N)$. Such cost functions can also be obtained as expectations of polynomial cost terms, or expectations of Taylor-expanded nonlinear cost terms.

The gPC-DDP algorithm

After expanding the random quantities and performing Galerkin projections as in section 2.3.1, we obtain for the gPC coefficients of the states:

$$\dot{\mathbf{X}}(t) = \mathbf{f}(\mathbf{X}(t), u(t), t), \tag{3.23}$$

with $\mathbf{X} := (x_{10}, ..., x_{1K}, ..., x_{ij}, ..., x_{nK})^\top \in \mathbb{R}^{n(K+1)}$ and $\dot{x}_{ij}(t) = \frac{\int_{\mathcal{D}} f_i(x(t,\xi),u(t),t,\xi)\phi_j(\xi)\rho(\xi)\mathrm{d}\xi}{\langle\phi_j,\phi_j\rangle}$. Polynomial Chaos theory allows us to estimate the moments of a stochastic process analytically [19]. More precisely, using the orthogonality of $\{\phi_j\}$ we get

$$\begin{aligned}(\hat{\mathcal{M}}_i)_1(t) = x_{i0}(t), \quad (\hat{\mathcal{M}}_i)_2(t) = \textstyle\sum_{j=1}^{K} x_{ij}^2(t)\langle\phi_j,\phi_j\rangle,\\ (\hat{\mathcal{M}}_i)_3(t) = \textstyle\sum_{j=0}^{K}\sum_{g=0}^{K} x_{ij}(t)x_{ig}(t)\big(\sum_{l=0}^{K} x_{il}(t)\langle\phi_j,\phi_g,\phi_l\rangle - 3x_{i0}(t)\big) + 2x_{i0}^3(t), \quad ...\end{aligned} \tag{3.24}$$

Here, $(\hat{\mathcal{M}}_i)_j$ denotes the gPC estimate for the j^{th} (central) moment of state x_i. It is easy to see that the above equations transform the moment-based problem (3.22) into its deterministic analogue:

$$\begin{aligned}\min_u \ & \int_{t_0}^{t_f} \boldsymbol{L}(\mathbf{X}, u, t)\mathrm{d}t + \boldsymbol{F}(\mathbf{X}(t_f), t_f)\\ \text{s.t.} \quad & \dot{\mathbf{X}}(t) = \mathbf{f}(\mathbf{X}, u, t), \quad \mathbf{X}(t_0) = \overline{\mathbf{X}}^0,\end{aligned} \tag{3.25}$$

where $\boldsymbol{L}(\mathbf{X}, u, t) := \mathcal{L}(\hat{\mathcal{M}}(\mathbf{X}(t)), u, t)$ and $\boldsymbol{F}(\mathbf{X}(t_f)) := \mathcal{F}(\hat{\mathcal{M}}(\mathbf{X}(t_f)), t_f)$. We proceed by deriving the Differential Dynamic Programming method for the discrete counterpart of (3.25). In what follows, $\boldsymbol{L}^k(\mathbf{X}^k, u^k) := \boldsymbol{L}(\mathbf{X}(t_k), u(t_k))\Delta t$ is the discrete approximation of the corresponding time integral in (2.2). Here, the problem will be constrained by discrete (in time) transition dynamics of the form: $\mathbf{X}^{k+1} = \mathbf{f}^k(\mathbf{X}^k, u^k)$. These can be derived from the continuous model in (3.25) either implicitly or explicitly (for example, the forward

Euler method reads $\mathbf{f}^k = \mathbf{X}^k + \Delta t \mathbf{f}(\mathbf{X}^k, u^k, t^k))$. Based on section 2.3.1, we can write:

$$V^k(\mathbf{X}^k) = \min_{u^k} \big[\boldsymbol{L}^k(\mathbf{X}^k, u^k) + V^{k+1}(\mathbf{X}^{k+1}) \big], \tag{3.26}$$

Our goal is to find (sub)optimal controls $\{u^k\}$ that minimize the right-hand side of (3.26). Let us define Q^k as

$$Q^k(\mathbf{X}^k, u^k) := \boldsymbol{L}^k(\mathbf{X}^k, u^k) + V^{k+1}(\mathbf{X}^{k+1}). \tag{3.27}$$

Next, given a nominal sequence $\{(\overline{\mathbf{X}}^k, \bar{u}^k)\}$, we will use a Taylor expansion to express the transition of state perturbations over time:

$$\begin{aligned} \delta\mathbf{X}^{k+1} \approx & \nabla_{\mathbf{x}}\mathbf{f}^k \delta\mathbf{X}^k + \nabla_u \mathbf{f}^k \delta u^k + \tfrac{1}{2}\big(\textstyle\sum_{i,j=1}^{\boldsymbol{n}} \nabla_{\mathbf{x}_i\mathbf{x}_j}\mathbf{f}^k \delta\mathbf{X}_i^k \delta\mathbf{X}_j^k + \\ & \textstyle\sum_{i,j=1}^{m} \nabla_{u_i u_j}\mathbf{f}^k \delta u_i^k \delta u_j^k + 2\sum_{i,j=1}^{\mathbf{n},m} \nabla_{\mathbf{x}_i u_j}\mathbf{f}^k \delta\mathbf{X}_i^k \delta u_j^k \big), \end{aligned} \tag{3.28}$$

where $\delta\mathbf{X}^k := \mathbf{X}^k - \overline{\mathbf{X}}^k$ and $\delta u^k := u^k - \bar{u}^k$ correspond to state and control deviations respectively.

By using (3.28), we can apply a quadratic expansion on the right-hand side of (3.27). It is straightforward to obtain the expression:

$$\begin{aligned} Q^k \approx & Q_0^k + (\delta\mathbf{X}^k)^\top Q_{\mathbf{x}}^k + (\delta u^k)^\top Q_u^k + \tfrac{1}{2}\big((\delta\mathbf{X}^k)^\top Q_{\mathbf{xx}}^k \delta\mathbf{X}^k + \\ & (\delta u^k)^\top Q_{uu}^k \delta u^k + (\delta u^k)^\top Q_{u\mathbf{x}}^k \delta\mathbf{X}^k + (\delta\mathbf{X}^k)^\top Q_{\mathbf{x}u}^k \delta u^k \big), \end{aligned} \tag{3.29}$$

where each derivative of Q above is properly defined. It can be found that the optimal controls will satisfy $\delta u_*^k = -(Q_{uu}^k)^{-1} Q_u^k - (Q_{uu}^k)^{-1} Q_{u\mathbf{x}}^k \delta\mathbf{X}^k$. Note that the solution above has been derived on the basis of quadratic approximations of the cost and dynamics. To this end, we will employ a line search parameter $\gamma \in (0, 1]$ and compute the updated control

inputs in an iterative fashion:

$$u^k_{(l+1)} = u^k_{(l)} - \gamma(Q^k_{uu,(l)})^{-1}Q^k_{u,(l)} - (Q^k_{uu,(l)})^{-1}Q^k_{u\mathbf{x},(l)}\delta\mathbf{X}^k_{(l)}. \tag{3.30}$$

Here, subscript (l) is associated with the l^{th} iteration of our scheme, at which the nominal controls correspond to: $\bar{u}^k \equiv u^k_{(l-1)}$; that is, the value of the decision variables at the previous iteration. The algorithm proceeds by computed a backpropagation scheme for the Q function and its derivatives. These expressions are omitted due to space limitations. Before concluding this subsection, we note that gPC-DDP requires the Jacobians and Hessians of the Polynomial Chaos-based dynamics, $\mathbf{f}^k$. These can be found in [34].

Incorporating a Variational Integrator

When we work with mechanical systems, we can use a VI for the propagation and linearization phases of gPC-DDP. Towards this goal, we provide explicit expressions for the associated Lagrangian function and non-conservative forces (e.g., control inputs, or dissipation). Consequently, Variational Integrators can be designed to obtain faithful discrete representations of stochastic systems.

Notice that any mechanical system will satisfy eqs. (2.17). In the presence of uncertain parameters $\xi \in \mathbb{R}^d$, one can use a gPC expansion on the coordinates q, v and p. That is

$$q_i(t) \approx \sum_{j=0}^{K} q_{ij}(t)\phi_j(\xi), \quad v_i(t) \approx \sum_{j=0}^{K} v_{ij}(t)\phi_j(\xi), \quad p_i(t) \approx \sum_{j=0}^{K} p_{ij}(t)\phi_j(\xi). \tag{3.31}$$

Now, define the concatenated vectors $\mathbf{Q} := (q_{10}, ..., q_{1K}, ..., q_{NK}) \in \mathbb{R}^{N(K+1)}$ and $\mathbf{V} := (v_{10}, ..., v_{1K}, ..., v_{NK}) \in \mathbb{R}^{N(K+1)}$. Let also $\hat{\mathbf{P}} := (\hat{p}_{10}, ..., \hat{p}_{1K}, ..., \hat{p}_{NK}) \in \mathbb{R}^{N(K+1)}$ be the set of unnormalized momentum coefficients, with $\{\hat{p}_{ij}\} := \{p_{ij}\langle\phi_j, \phi_j\rangle\}$. Consider the following lemma:

Lemma 1. *Consider a mechanical system with uncertain parameters ξ, and let L, F denote*

its Lagrangian function and set of non-conservative forces, respectively. Suppose also that its position, velocity and momentum coordinates can be expanded as in (3.31)*. Then, the system of gPC coefficients* $(\mathbf{Q}, \mathbf{V}, \hat{\mathbf{P}})$ *will satisfy eqs.* (2.17) *with*

$$\hat{L}(\mathbf{Q}, \mathbf{V}) := \int_{\mathcal{D}} L\rho(\xi)\mathrm{d}\xi \tag{3.32}$$

being the associated Lagrangian function, and

$$\hat{\mathbf{F}} := (\hat{F}_{10}, ..., \hat{F}_{1K}, ..., \hat{F}_{NK}) \in \mathbb{R}^{N(K+1)}, \quad \hat{F}_{ij}(\mathbf{Q}, \mathbf{V}, u) := \int_{\mathcal{D}} F_i \phi_j(\xi)\rho(\xi)\mathrm{d}\xi \tag{3.33}$$

being the non-conservative forces.

Proof 2. *For Lagrangian systems, eqs.* (2.17) *must be satisfied. Plugging* (3.31) *in* (2.17) *and performing Galerkin projection gives*

$$v_{ij} = \dot{q}_{ij}, \tag{3.34}$$

$$\hat{p}_{ij} = \textstyle\int_{\mathcal{D}} \frac{\partial L}{\partial v_i}\phi_j(\xi)\rho(\xi)\mathrm{d}\xi, \tag{3.35}$$

$$\dot{\hat{p}}_{ij} = \textstyle\int_{\mathcal{D}} \frac{\partial L}{\partial q_i}\phi_j(\xi)\rho(\xi)\mathrm{d}\xi + \int_{\mathcal{D}} F_i\phi_j(\xi)\rho(\xi)\mathrm{d}\xi, \tag{3.36}$$

for $i = 1, ..., N$ *and* $j = 0, ..., K$*. Now, define the functions* $\hat{L}$*,* $\hat{F}_{ij}$ *as in equation* (3.32) *and* (3.33)*, respectively. Since the gPC coefficients are deterministic, one can obtain*

$$\frac{\partial \hat{L}}{\partial v_{ij}} = \textstyle\int_{\mathcal{D}} \frac{\partial L}{\partial v_i}\frac{\partial v_i}{\partial v_{ij}}\rho(\xi)\mathrm{d}\xi = \int_{\mathcal{D}} \frac{\partial L}{\partial v_i}\phi_j(\xi)\rho(\xi)\mathrm{d}\xi, \tag{3.37}$$

where the first equality is due to chain rule, and the second equality is due to (3.31)*. Equations* (3.35) *and* (3.37) *yield*

$$\hat{p}_{ij} = \frac{\partial \hat{L}}{\partial v_{ij}}. \tag{3.38}$$

In a similar manner, one can show that

$$\frac{\partial \hat{L}}{\partial q_{ij}} = \int_{\mathcal{D}} \frac{\partial L}{\partial q_i} \frac{\partial q_i}{\partial q_{ij}} \rho(\xi) \mathrm{d}\xi = \int_{\mathcal{D}} \frac{\partial L}{\partial q_i} \phi_j(\xi) \rho(\xi) \mathrm{d}\xi. \tag{3.39}$$

Combining (3.33)*,* (3.36) *and* (3.39) *gives*

$$\dot{\hat{p}}_{ij} = \frac{\partial \hat{L}}{\partial q_{ij}} + \hat{F}_{ij}. \tag{3.40}$$

The conclusion is made by comparing (3.34)*,* (3.38) *&* (3.40) *with equation* (2.17)*.*■

Finally, we note that will require the Jacobians and Hessians of $\hat{L}^k$ and $\hat{\mathbf{F}}^{k\pm}$. Below we give an expression for $D_1 D_1 \hat{L}^k$. The remaining quantities can be determined similarly.

Proposition 1. *Let* $\hat{L}^k$ *be defined as in* (3.32)*. Then*

$$D_1 D_1 \hat{L}^k = \int_{\mathcal{D}} \big(D_1 D_1 L_d(q^k, q^{k+1}, \xi) \otimes \big(\Phi(\xi) \otimes \Phi(\xi)^\top \big) \big) \rho(\xi) \mathrm{d}(\xi), \tag{3.41}$$

Notice that the propagation and linearization phases require computing integrals of the form $\int_{\mathcal{D}} \mathsf{f}(\xi)\rho(\xi)\mathrm{d}\xi$, where $\mathsf{f} \in L^2_\rho$ (see, e.g., (3.41)). These quantities can rarely be calculated analytically; except if linear or sufficiently simple dynamics are considered. For our simulations, we employed *Gaussian quadrature*. Given the density function $\rho(\xi)$, this method selects nodes $\{z_{l_i}\}$ and weights $\{w_{l_i}\}$ such that

$$\int_{\mathcal{D}_1} \dots \int_{\mathcal{D}_d} \mathsf{f}(\xi)\rho(\xi)\mathrm{d}\xi \approx \sum_{l_1=1}^{l_{gq}} \dots \sum_{l_d=1}^{l_{gq}} w_{l_1} \dots w_{l_d} \mathsf{f}(z_{l_1}, \dots, z_{l_d}), \quad \xi \in \mathbb{R}^d, \quad z_{l_i} \in \mathbb{R}. \tag{3.42}$$

Here, $\{z_{l_i}\}$ and $\{w_{l_i}\}$ are obtained by solving an eigenvalue decomposition problem [25]. The above formula holds with equality when the integrand is a polynomial of degree less than or equal to $2l_{gq} - 1$. Since we consider smooth dynamics in this paper, the particular integration method is expected to be highly accurate. Lastly, observe that the full-tensor expression in (3.42) uses l_{gq}^d function evaluations. When d is large, one can use sparse

quadrature formulas to reduce the number of nodes, while retaining sufficient precision. More details can be found in [35].

Remark 1. *gPC-DDP alternates between propagating nominal rollouts from the stochastic system, and computing (sub)optimal control deviations about them. Hence, dynamics constraints are inherently satisfied by our optimizer. As shown in our simulations section, this is a major benefit, especially when augmented state-spaces under Polynomial Chaos expansions have to be handled.*

Simulations

Duffing oscillator: We will present numerical results after applying gPC-DDP on the dynamics of the Duffing oscillator [32]. The dynamics are given by

$$\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\lambda x_1 - \frac{1}{4} x_2 - x_1^3 + u \end{pmatrix}. \tag{3.43}$$

The uncertainty of our system will lie in the parameter λ, as well as the initial state $x_1(t_0)$. We consider these quantities to be normally distributed, with $\lambda \sim \mathcal{N}(\mu_\lambda, \sigma_\lambda^2)$ and $x_1(t_0) \sim \mathcal{N}(\mu_1^0, (\sigma_1^0)^2)$. Based on (2.1), Hermite polynomials will be employed. The first few (unnormalized) Hermite polynomials are given by [19]

$$\phi_0(\mathsf{x}) = 1, \quad \phi_1(\mathsf{x}) = \mathsf{x}, \quad \phi_2(\mathsf{x}) = \mathsf{x}^2 - 1, ..., \quad \mathsf{x} \in \mathbb{R}. \tag{3.44}$$

Furthermore, some important expressions are

$$\rho(\mathsf{x}) = \frac{1}{\sqrt{2\pi}} \exp^{-\mathsf{x}^2/2}, \quad \langle \phi_i, \phi_j \rangle = \delta_{ij} i!,$$

$$\langle \phi_i, \phi_j, \phi_g \rangle = \begin{cases} \frac{g!j!i!}{(s-g)!(s-j)!(s-i)!}, & i, j, g \text{ even } \& \ s \geq \max(i, j, g) \\ 0, & \text{otherwise} \end{cases}$$

where $s = \frac{i+j+g}{2}$. Now, λ and $x_1(t_0)$ can be fully described as

$$\lambda(\xi^p) = \mu_\lambda \phi_0^p(\xi^p) + \sigma_\lambda \phi_1^p(\xi^p), \quad x_1^0(\xi^0) = \mu_1^0 \phi_0^0(\xi^0) + \sigma_1^0 \phi_1^0(\xi^0), \tag{3.45}$$

with $\xi^p, \xi^0 \sim \mathcal{N}(0,1)$. In this setting, the states are influenced by $\xi = (\xi^p, \xi^0) \in \mathbb{R}^2$. Applying the gPC expansion on the state vector yields

$$x_1(t,\xi) \approx \sum_{j=0}^K x_{1j}(t)\phi_j(\xi), \quad x_2(t,\xi) \approx \sum_{j=0}^K x_{2j}(t)\phi_j(\xi), \tag{3.46}$$

with $\phi_j(\xi) = \phi_{j_1}^p(\xi^p)\phi_{j_2}^0(\xi^0), \quad 0 \leq j_1 + j_2 \leq r$ (r being the maximum total order of each ϕ_j). Since $x_2(t_0)$ is deterministic, we take $x_{20}(t_0) = x_2(t_0)$ and $x_{2j}(t_0) = 0$ for $j > 0$. Plugging (3.45), (3.46) in (3.43), and performing Galerkin projection gives

$$\begin{aligned}
&\dot{x}_{1l} = x_{2l} \\
&\dot{x}_{2l} = \tfrac{1}{\langle \phi_l, \phi_l \rangle}\big(- \textstyle\sum_{i=0}^K x_{1i}\big(\mu_\lambda \langle \phi_0^p, \phi_l, \phi_i \rangle + \sigma_\lambda \langle \phi_1^p, \phi_l, \phi_i \rangle\big) - \\
&\tfrac{1}{4} x_{2l}\langle \phi_l, \phi_l \rangle - \textstyle\sum_{i=0}^K \sum_{g=0}^K \sum_{j=0}^K x_{1i}x_{1g}x_{1j}\langle \phi_l, \phi_i, \phi_g, \phi_j \rangle + \langle \phi_l \rangle u\big), \quad l = 0, ..., K.
\end{aligned} \tag{3.47}$$

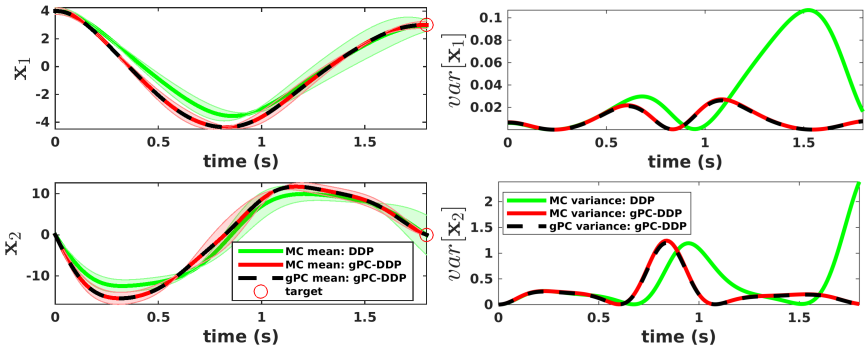
For our simulations we pick: $\mu_\lambda = 3$, $\sigma_\lambda = 0.1$, $\mu_1^0 = 4$, $\sigma_1^0 = 0.08$ and $x_2^0 = 0$. Our task will be to reach the target state $x^{goal} = (3,0)^\top$ in $t_f = 1.8$ s. The running cost is set to $\boldsymbol{L} = \frac{1}{2}0.01u^2$, while the terminal cost term is selected as $\boldsymbol{F} = \frac{1}{2}\sum_{i=1}^2((x_{i0}(t_f) - x_i^{goal})^2 s_{f_{i0}} + \sum_{l=1}^K x_{il}^2(t_f) s_{f_{ij}})$, with $s_{f_{10}} = s_{f_{20}} = 400$, and $s_{f_{1j}} = 300$, $s_{f_{2j}} = 100$ for each $j > 0$. Hence, gPC-DDP will penalize trajectories with (i) expected terminal states far from the desired ones, (ii) high terminal variance. For the gPC expansions, we selected $r = 3$, resulting in an $\boldsymbol{n} = 20$ state vector (see (3.46)). Due to space limitations, we will not show how the accuracy of the gPC approximation changes with the number of expansion terms, K. In general, the estimated moments get closer to the actual ones as K increases, with the convergence rate depending on the quantity to be approximated (see [5], [31], [19]).

The results of our algorithm are displayed in 3.9. Therein, we have used a forward Euler discretization of (3.47). Moreover, gPC-DDP was initialized with zero controls, while the convergence criterion was set to $J_{(i)} \leq 10^{-8}$ (here, $J_{(i)}$ is the cost at iteration i).

In Figure 3.9 we make a qualitative comparison between gPC-DDP and the original Differential Dynamic Programming method. gPC-DDP is capable of guiding the expected states to the target, while also minimizing the terminal variance. In Figure 3.10a we provide further details about the behavior of gPC-DDP. Therein, we show the effect the linearization order has on the speed of convergence (plots correspond to control-dependent running cost). In particular, we compare between using only the first-order terms from (3.28), as opposed to the full expansion. We propose switching between the two schemes, depending on the progress of gPC-DDP. Specifically, we employ a first-order expansion during the first iterations, and shift to a quadratic model when the change in the cost gets under a prespecified threshold. To proceed, Figure 3.10b includes the results of an off-the-shelf optimization solver as benchmark comparison. Therein, we employed MATLAB's built-in SQP implementation for solving problem (3.25).

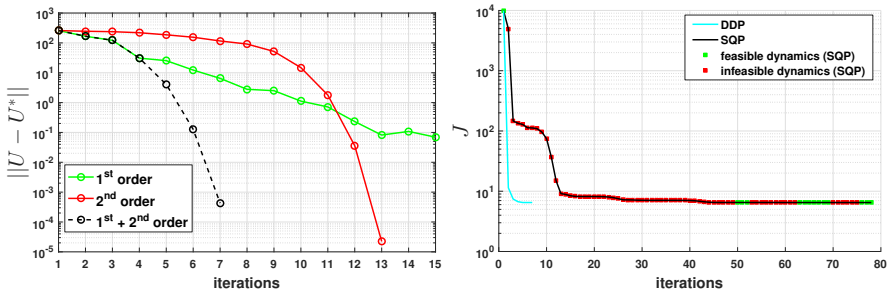
To show the importance of our variational integration scheme, we compare the (locally) optimal trajectories given by gPC-DDP & VI-gPC-DDP for varying time steps, Δt. In general, a properly discretized model will behave similarly to its continuous counterpart, even for relatively large values of Δt. In that case, different time-step selections would only induce small deviations in the solutions. Figure 3.11 shows the gPC mean estimates after implementing each of the aforementioned settings. It can be easily deduced that utilizing our Variational Integrator highly reduces the impact of Δt on the obtained trajectories.

Quadrotor: The state vector considered here is $x := (\chi^\top, \eta^\top, \dot{\chi}^\top, \dot{\eta}^\top)^\top \in \mathbb{R}^{12}$, where $\chi := (\mathsf{x}, \mathsf{y}, \mathsf{z})^\top \in \mathbb{R}^3$, $\eta := (\phi, \theta, \psi)^\top \in \mathbb{R}^3$ denote its position and Euler angles, respectively (see (3.12a) for an illustration). Regarding its internal parameters (see (3.12b) and (3.12c) for values thereof): l is the distance between the rotors and center of mass, m is the mass, g is the gravitational acceleration, $\mathbb{I} := \text{diag}(\mathbb{I}_\mathsf{x}, \mathbb{I}_\mathsf{y}, \mathbb{I}_\mathsf{z})$ is the inertia matrix and G_{tr},



(a) State trajectories (b) State variance

Figure 3.9: Duffing oscillator: Comparison between gPC-DDP and deterministic DDP (the latter applied on the mean parameter values). On the left-hand side, the Monte Carlo and gPC estimates of the expected states are illustrated, along with $\pm 3\sigma$ of sampled trajectories (green and red shaded areas). The right-hand side depicts the variance (gPC and MC estimates) of the state trajectories obtained by each algorithm.



(a) gPC-DDP covergence rates (b) gPC-DDP covergence rates

Figure 3.10: Duffing oscillator: The left figure shows how different expansion orders of dynamics affect the convergence rate of gPC-DDP. The green and red lines correspond to a first, and second-order approximation scheme, respectively. In contrast, the dashed black line employs the quadratic terms only from the fourth iteration, reaching thus the solution in fewer steps. The right figure compares gPC-DDP to an off-the-shelf SQP solver, and depicts the cost value per iteration for each method. Both approaches reached the same solution. However, SQP was approximately 70 times slower, and required more iterations to converge.

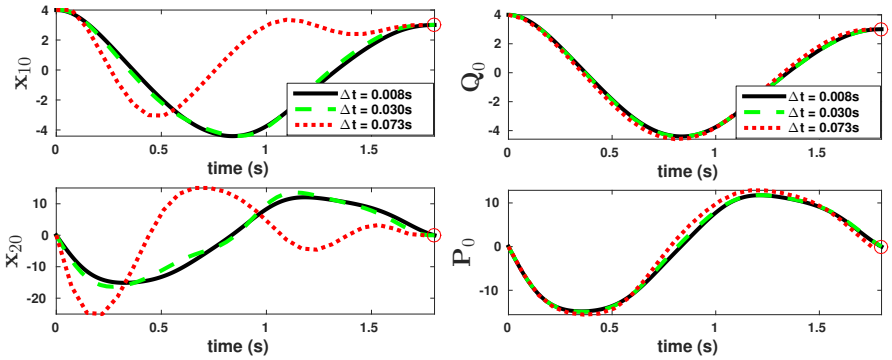


Figure 3.11: Duffing oscillator: Comparison between gPC-DDP and VI-gPC-DDP for different step sizes, Δt. The former method usedi an explicit Euler scheme to propagate and linearize the gPC-based dynamics. The gPC mean estimates are able to reach the target for all cases. However, VI-gPC-DDP is much more insensitive to the selection of Δt.

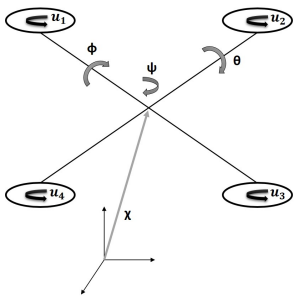
G_{rot} are constants whose values depend on the air density and rotor shape.

We consider the case where G_{tr} and G_{rot} are uniformly distributed, with known upper and lower bounds. Now, based on 2.1, we will employ Legendre polynomials for our gPC expansions (for some useful expressions refer to [5]). We picked the maximum order of our polynomials to be $r = 2$. This resulted in an $\boldsymbol{n} = 72$ state vector. Note also that in this example, the propagation and linearization phases of gPC-DDP cannot be computed analytically. Hence, we utilized Gaussian quadrature, for which we employed $l_{gq} = 3$ quadrature nodes (see (3.42)).

Dealing with stochastic disturbances

To handle stochastic disturbances within gPC-DDP, we use the Karhunen-Loéve (KL) method from equation (2.15). Hence, we decompose a stochastic process into a series of products involving deterministic functions of time and random variables. In this way, Galerkin projections can be applied to determine the dynamics of the gPC coefficients, as well as the stochastic forces required for using VI's. For more details see [33].

The aforementioned plots were generated through an explicit Euler discretization method. Let us now test the effect of our VI on this high-dimensional problem. In particular, we consider the following scenario: We begin by solving the optimization problem for differ-



(a) Illustration

g	$9.81\ m/s^2$
m	$1\ kg$
l	$0.24\ m$
$\mathbb{I}_x$	$8.1 \cdot 10^{-3}\ kgm^2$
$\mathbb{I}_y$	$8.1 \cdot 10^{-3}\ kgm^2$
$\mathbb{I}_z$	$14.2 \cdot 10^{-3}\ kgm^2$

(b) Deterministic parameters

	min	max
G_{tr}	$2.85 \cdot 10^{-5}$	$2.95 \cdot 10^{-5}$
G_{rot}	$1.05 \cdot 10^{-6}$	$1.15 \cdot 10^{-6}$

(c) Uniformly distributed, stochastic parameters

Figure 3.12: The quadrotor model - details.

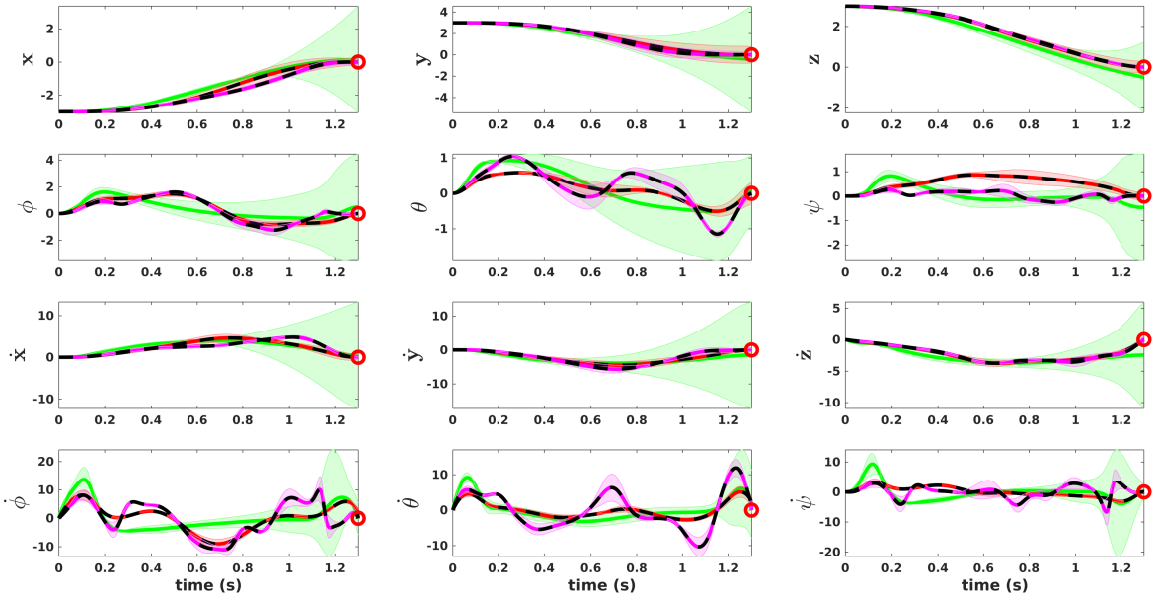


Figure 3.13: Quadrotor: Comparison between gPC-DDP (red & magenta) and deterministic DDP (green). Regarding the former, two settings are considered with different uncertainty penalization levels - red: low, magenta: high. Solid lines represent Monte Carlo mean estimates, while black dashed lines represent gPC mean estimates. The colored shaded areas correspond to $\pm 3\sigma$ of trajectories sampled under the different control sequences. Each algorithm was initialized with zero controls (units - (x, y, z): m, (ϕ, θ, ψ): rad, $(\dot{x}, \dot{y}, \dot{z})$: m/s, $(\dot{\phi}, \dot{\theta}, \dot{\psi})$: rad$/s$).

ent time steps, Δt. Then, we evaluate each solution by plugging the corresponding controls

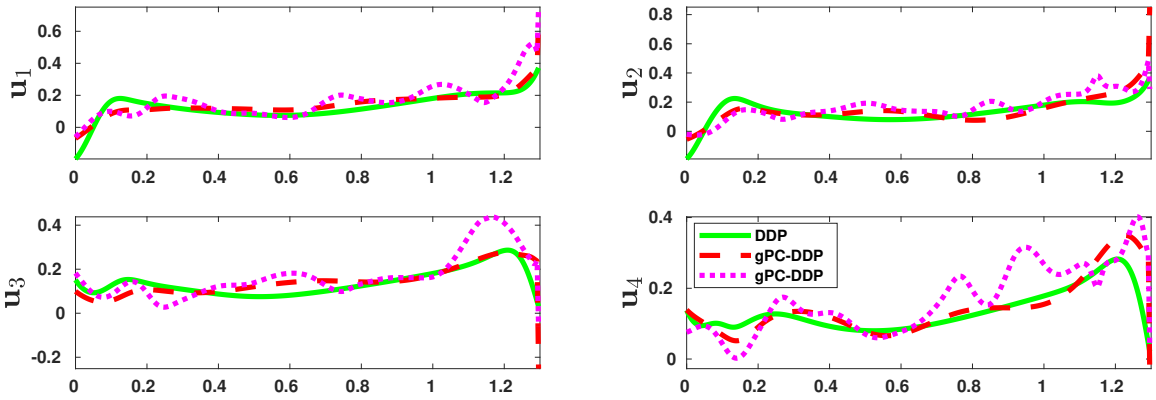


Figure 3.14: Quadrotor: Controls obtained by deterministic DDP (green) and gPC-DDP for different uncertainty penalization levels (red: low, magenta: high).

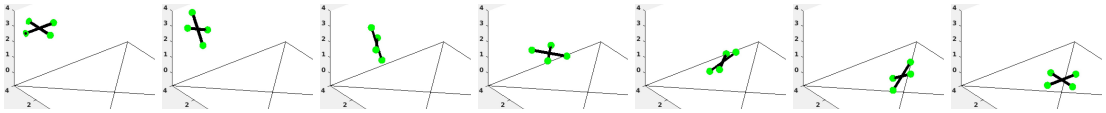
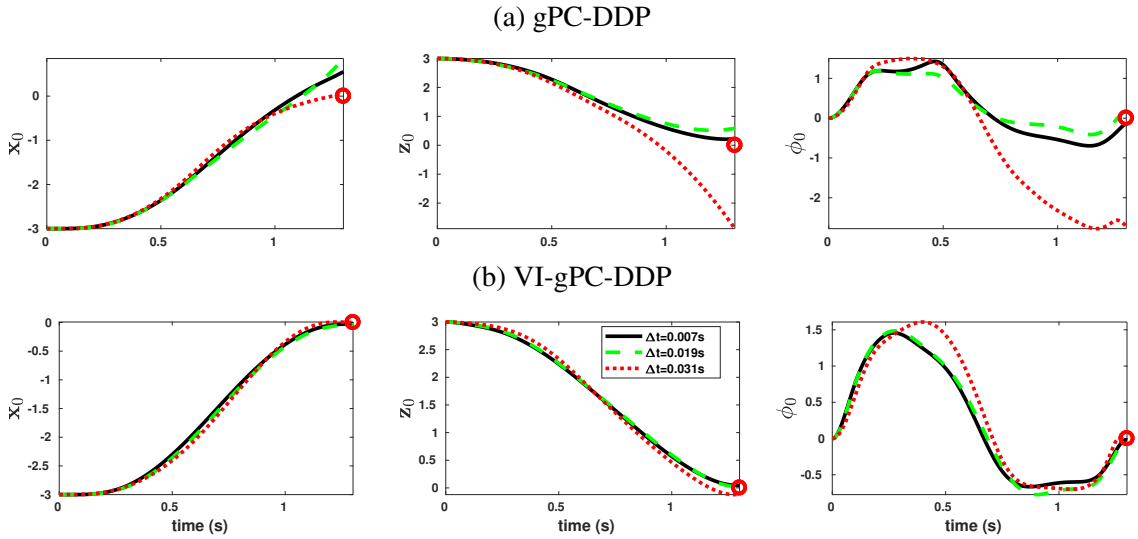


Figure 3.15: Quadrotor: Instances of the (sub)optimal, mean state trajectory obtained by gPC-DDP.

into the actual, continuous[1] dynamics (3.23). Figure 3.16c compares three of the gPC mean estimates between an Euler-based gPC-DDP approach, and VI-gPC-DDP. It is deduced that naive discretizations can highly degrade the planning procedure, especially when relatively large time steps are employed.

Being able to use coarse discretizations can also reduce the computational complexity of our algorithm. Figure 3.17 compares VI-gPC-DDP to gPC-DDP in terms of the average time required for their propagation phases. Notice that the variational integration scheme will be slower for similar step sizes, since one has to solve the DEL equations implicitly. Nevertheless, when the discrete horizon is sufficiently decreased, the overall elapsed time will be reduced.

[1]Here, we simulate continuous-like dynamics by applying the forward Euler method on (3.23) with a very small discretization step ($\Delta t = 0.001$).



(c) Quadrotor: Comparison between gPC-DDP & VI-gPC-DDP. The legend shows the discretization step that we used when implementing each scheme. The plots were obtained by applying the (sub)optimal controllers on the continuous gPC-based dynamics.

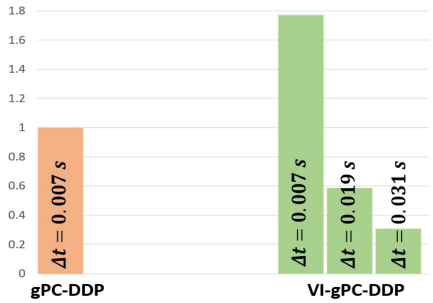


Figure 3.17: Quadrotor: Average elapsed time for the propagation phase of VI-gPC-DPP under different time steps, Δt. The results are normalized with respect to gPC-DDP's corresponding time, for $\Delta t = 0.007s$. Utilizing the developed VI allows for coarse discretizations and, therefore, can reduce computational complexity.

Convergence properties of gPC-DDP

The analysis presented here goes along the lines of [36] and [37]. However, these works relied on certain simplifications to obtain their results. For instance, [36] did not consider running costs in its formulation, while [37] dealt only with scalar dynamic systems. In contrast, we will address the generic optimal control problem in (3.25). Here, $U \in \mathbb{R}^{(K_f-1)m}$ denotes an entire control sequence (i.e., $U := ((u^0)^\top, (u^1)^\top, ..., (u^{K_f-1})^\top)$). Also, we consider the next set of assumptions.: (i) The dynamics and cost functions of (3.25) are differentiable up to the third order over a compact convex set $\mathcal{R} \in \mathbb{R}^{\boldsymbol{n}+m}$, (ii) $((\boldsymbol{X}^k)^\top, (u^k)^\top)^\top \in \mathcal{R}, \forall k$, (iii) $Q^k_{uu} > 0$ for each $k = 0, 1, ...$ The convergence properties of gPC-DDP are established by the next theorems and corollaries. In particular, (3.49) implies that gPC-DDP will converge to a solution when the line-search parameter, γ, is sufficiently small. Furthermore, (3.50) points to a (locally) quadratic convergence rate of the algorithm.

Theorem 5. *Consider the discrete-time optimal control problem*

$$\begin{aligned} &\min_{u^0, u^1, ..., u^{K_f-1}} \quad \boldsymbol{J} \\ &\textit{s.t.} \quad \mathbf{X}^{k+1} = \mathbf{f}^k(\mathbf{X}^k, u^k), \quad \mathbf{X}^0 = \overline{\mathbf{X}}^0, \end{aligned} \tag{3.48}$$

where $\boldsymbol{J} := \sum_{k=0}^{K_f-1} \boldsymbol{L}^k(\mathbf{X}^k, u^k) + \boldsymbol{F}(\mathbf{X}^{K_f})$. *Let also* $\delta U_* := ((\delta u^0_*)^\top, (\delta u^1_*)^\top, ...)$ *include the control updates of gPC-DDP, with* $\delta u^k_* := -\gamma(Q^k_{uu})^{-1}Q^k_u - (Q^k_{uu})^{-1}Q^k_{u\boldsymbol{x}}\delta\mathbf{X}^k$. *Then:*

$$(\nabla_U \boldsymbol{J}|_{\bar{U}})^\top \delta U_* = -\gamma \textstyle\sum_{k=0}^{K_f-1} (Q^k_u)^\top (Q^k_{uu})^{-1} Q^k_u + O(\gamma^2). \tag{3.49}$$

Corrolary 1. *Given any bounded* $\mathbf{X}^0$*, the gPC-DDP algorithm will converge to a stationary solution of* (3.48)*.*

Theorem 6. *Let* $U_{(l)}$ *denote the controls obtained at iteration* l*, and* $U_\star$ *be the stationary*

point gPC-DDP converges to. Then, there exist $c > 0$ *and* $l_\star > 0$*, such that:*

$$||U_{(l)} - U_\star|| \leq c||U_{(l-1)} - U_\star||^2, \quad \forall l \geq l_\star. \tag{3.50}$$

Therefore, the convergence rate will be locally quadratic.

Algorithmically, one can account for the positive definiteness assumption by setting [10]: $Q_{uu}^k \leftarrow Q_{uu}^k + \theta I$. Here, $\theta > 0$ is a regularization parameter that enforces the required condition for all time instances.

3.2.2 Chance-constrained control

The developed frameworks allow us to consider the control of stochastic systems under chance constraints. Due to the intractability of such problems, chance constraints are often replaced with deterministic surrogates derived in terms of either samples or moments of the probability distribution of the state [38]. A common approach is via conservative approximations with a collection of individual chance constraints based on a risk allocation derived from Boole's inequality. The individual chance constraints can then be approximated using the distributionally robust Cantelli-Chebyshev inequality, which replaces each chance constraint with an expression in terms of the mean and variance of the state.

Discussing such representations of chance constraints in detail goes beyond the scope of this thesis. We will instead consider constraints affine in the state of the system:

$$Hx + d \leq 0 \tag{3.51}$$

and assume at each time instant that each state follows a Gaussian distribution, $x_i^k \sim N(\mu_i^k, \Sigma_i^k)$. Based on the latter assumption, we can have the following confidence intervals

$$\mu_i^k - \kappa_\delta \sqrt{\Sigma_i^k} \leq x_i^k \leq \mu_i^k + \kappa_\delta \sqrt{\Sigma_i^k} \tag{3.52}$$

with probability $\rho(\kappa_\delta)$, where κ_δ is a hyperparameter that affects $\rho(\kappa_\delta)$. For example, 95% confidence interval is attained when $\kappa_\delta = 2$.

Now, based on this fact, the constraint in (3.51) will be satisfied (at least) with probability $\rho(\kappa_\delta)$, if we enforce the following moment-based constraint:

$$\sum_i H_i\big(\mu_i^k + \text{sign}(H_i)\kappa_\delta\sqrt{\Sigma_i^k}\big) + d \leq 0. \tag{3.53}$$

When gPC expansions are used, the mean and variance can be approximated via the series coefficients. Hence, an estimation to (3.53) is

$$\sum_i H_i\big(x_{i0}^k + \text{sign}(H_i)\kappa_\delta\sqrt{\sum_{j=1}^{k}(x_{ij}^k)^2}\big) + d \leq 0. \tag{3.54}$$

Therefore, the methodologies of the previous sections can be used to control stochastic systems under constraints of the form (3.54)[2].

Simulations

We compare the behavior of the KKT+AL DDP algorithm applied on the Duffing oscillator system for different values of κ_δ. The behavior of the unconstrained version of the algorithm is also provided. The results are given in figures 3.19, 3.18, 3.20.

[2]Since the square root is not differentiable at 0, a cubic polynomial was used to approximate $\sqrt{y}$ when $|y| \leq \zeta$, for some small positive number ζ, with smooth first and second-order derivatives at $y = \zeta$. In this way, the cost function and constraints are twice differentiable and DDP-based trajectory optimization can be applied.

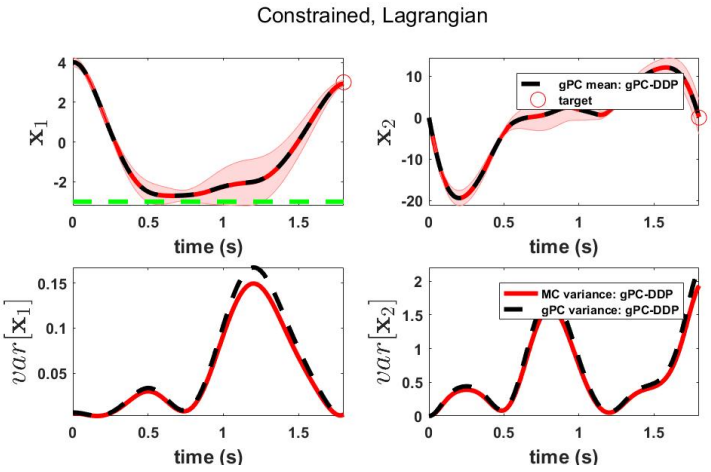


Figure 3.18: Duffing oscillator: Chance-constrained gPC-DDP with $\kappa_\delta = 1$.

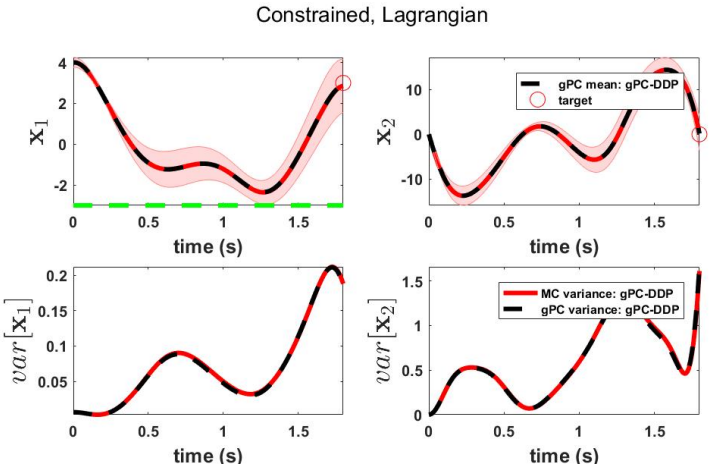


Figure 3.19: Duffing oscillator: Chance-constrained gPC-DDP with $\kappa_\delta = 3$.

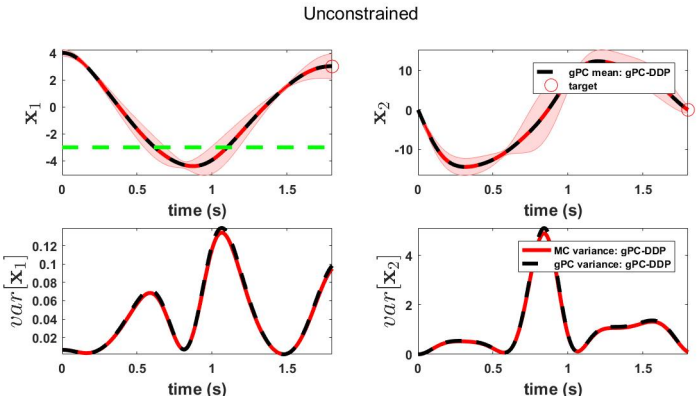


Figure 3.20: Duffing oscillator: Unconstrained gPC-DDP.

3.3 Control of systems with unknown dynamics - a model-based reinforcement learning approach

3.3.1 Unconstrained Reinforcement Learning

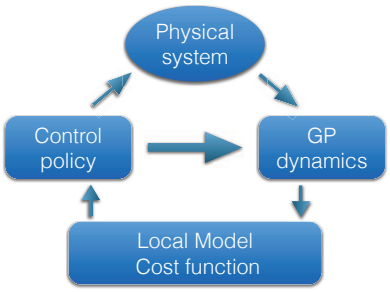
In contrast to the previous section, we will consider here systems with unknown form of dynamics. We will thus rely on collecting data from the agent and its environment to learn transition dynamics models. A nonparametric, ML-based approach will be used which will rely on Gaussian processes (GPs).

Over last few years, GP-based control and reinforcement learning (RL) algorithms have increasingly drawn more attention in control theory and machine learning fields. For instance, the works by Rasmussen and Kuss is one of the first GP-based RL algorithm [39]. More recently, Deisenroth et al. proposed model-based policy search using GPs [2]. These works have demonstrated the remarkable applicability of GP-based control and RL methods in robotics and autonomous systems. By incorporating Bayesian learning into DDP, in this work we propose a probabilistic trajectory optimization approach, called Probabilistic Differential Dynamic Programming (PDDP).

Key features

The major characteristics of PDDP can be summarized as follow:

- The PDDP learning framework takes advantages of previous works in GP inference and features data efficiency that is comparable to the state-of-the-art method.
- PDDP is derived based on local trajectory optimization via successive forward-backward sweeps. This highly efficient scheme leads to significant improved computational efficiency compared with related methods.
- PDDP can incorporate prior model knowledge from physical-based models into the learning framework. This results in a framework using semiparametric model learning and inference. Our method is more general than previous related works in semi-



PDDP	PILCO
• No required policy parameterization	• Pre-specified policy parameterization
• Self-contained optimizer	• Extra optimization solver required (e.g., BFGS)
• Policy improvement at each time step takes into account future steps.	• Policy improvement is time-independent
• Close-to-PILCO data efficiency	• State-of-the-art data efficiency
• Significantly improved computational efficiency	• Very high computational demand

parametric model learning. More precisely the model parameters do not have to be linear in the basis functions.

- PDDP is able to perform risk-sensitive learning by using the predicted cost distribution as the optimization criterion.

- PDDP can be applied for trajectory optimization tasks under unknown or uncertain dynamics using a very small number of interactions with the real physical systems.

Description of method

In order to incorporate dynamics uncertainty in trajectory optimization, we use analytic approximate inference (moment matching approach) [39, 2] to predict state distribution (belief) over trajectories. One-step prediction of distribution $x_{k+1} \sim \mathcal{N}(\mu_{k+1}, \Sigma_{k+1})$ can be written as follow

$$\mu_{k+1} = \mu_k + \mathrm{d}\mu_k, \qquad \Sigma_{k+1} = \Sigma_k + \mathrm{d}\Sigma_k + \mathrm{Cov}_{f,x_k}[x_k, \mathrm{d}x_k] + \mathrm{Cov}_{f,x_k}[\mathrm{d}x_k, x_k]. \quad (3.55)$$

The above equations can be simplified as $z_{k+1} = F(z_k, u_k)$. Define the control and state variations $\delta z_k = z_k - \bar{z}_k$ and $\delta u_k = u_k - \bar{u}_k$. We linerize the above model around a nominal trajectory: $\delta z_{k+1} = mF_k^z \delta z_k + F_k^u \delta u_k$, where the Jacobian matrices F_k^x and F_k^u are specified in [40]. All derivatives are computed analytically. In PDDP we use the cost function as the expectation of a quadratic cost function $\mathcal{L}(z_k, u_k) = \mathbb{E}_x\Big[\mathcal{L}(x_k, u_k)\Big] =$ trace$(Q\Sigma_k) + (\mu_k - x_k^{goal})^\top Q(\mu_k - x_k^{goal}) + u_k^\top R u_k$. The analytic derivatives of the cost

w.r.t both u and z can be easily obtained. Similarly to gPC-DDP, we create a quadratic local model of the value function by expanding the Q-function up to the second order

$$Q_k(z_k + \delta z_k, u_k + \delta u_k) \approx Q_k^0 + Q_k^z \delta z_k + Q_k^u \delta u_k + \frac{1}{2} \begin{bmatrix} \delta z_k \\ \delta u_k \end{bmatrix}^\top \begin{bmatrix} Q_k^{zz} & Q_k^{zu} \\ Q_k^{uz} & Q_k^{uu} \end{bmatrix} \begin{bmatrix} \delta z_k \\ \delta u_k \end{bmatrix}, \tag{3.56}$$

where the superscripts of the Q-function indicate derivatives. We use this superscript rule for $\mathcal{L}$ and V as well. Next we compute the local variations in control $\delta \hat{u}_k$ that maximize the Q-function

$$\delta \hat{u}_k = \arg\min_{u_k} \Big[Q_k(z_k + \delta z_k, u_k + \delta u_k) \Big] = -(Q_k^{uu})^{-1} Q_k^u - (Q_k^{uu})^{-1} Q_k^{uz} \delta z_k. \tag{3.57}$$

The optimal control can be found as $\hat{u}_k = \bar{u}_k + \delta \hat{u}_k$ and the quadratic expansion of the value function is propagated backward in time. The new policy is used to generate a new nominal trajectory via forward propagation. This backward-forward scheme is applied iteratively until convergence. We perform rollouts using the learned controller on the true system, obtain new sample data to update the GP dynamics model, and perform optimization initialized with the new trajectory. We repeat this trial and optimization scheme until the task is learned. The detailed algorithm can be found in [40]. The PDDP approach can be extended to take advantage of prior model knowledge such as basis function from physics-based modeling as well as incorporate risk-sensitive performance criteria [40]. We also notice that an analysis similar to gPC-DDP can be carried for PDDP as well.

Results and discussion

We evaluated the PDDP framework using two nontrivial simulated examples: i) cart-double inverted pendulum swing-up (CDIP); ii) six-link robotic arm reaching and iii) quadrotor target reaching. Performances were compared with PILCO [2].

As shown in Fig.3.22, in both tasks, PDDP performs slightly worse than PILCO in

Algorithm 3 : Probabilistic Differential Dynamic Programming for reinforcement learning

Initialization: Apply random or pre-specified control inputs to the physical system. Collect data.
repeat ▷ *Main reinforcement learning loop*
  Model learning: Learn GP hyper-parameters $\boldsymbol{\theta}$ by evidence maximizatio given sample data. Optimize dynamics model parameters if necessary.
  repeat ▷ *Probabilistic trajectory optimization loop*
    Local approximation: Obtain linear approximation of the belief dynamics and quadratic approximation of the cost along a nominal trajectory $(\bar{\mathbf{b}}_k, \bar{\mathbf{u}}_k)$.
    Backward sweep: Compute the approximation of the value function and obtain optimal policy for control correction $\delta\hat{\mathbf{u}}_k$.
    Forward sweep: Update control $\bar{\mathbf{u}}_k$ and perform approximate inference to obtain a new nominal trajectory $(\bar{\mathbf{b}}_k, \bar{\mathbf{u}}_k)$.
  until Termination condition is satisfied
  return Optimal trajectory $(\bar{\mathbf{b}}_k, \bar{\mathbf{u}}_k)$ and control policy
  Interaction: Apply the optimized control policy to the system along the optimized trajectory $\bar{\mathbf{u}}_k$ and record data. Additional rollouts can be generated using variations of the learned controller.
until Task learned
Return: Optimal trajectory and control policy

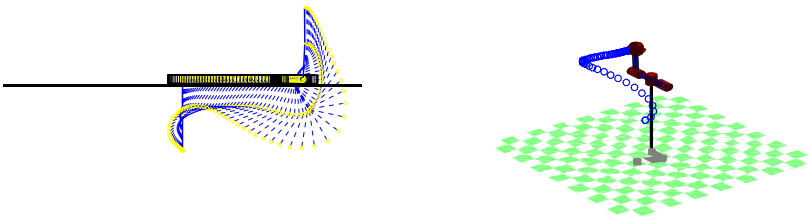


Figure 3.21: CDIP and Puma-560 tasks.

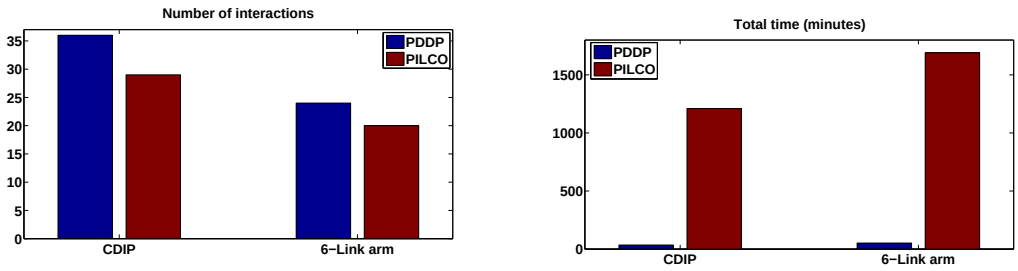
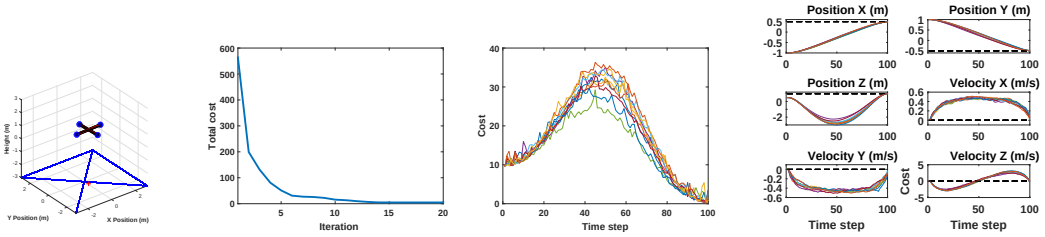


Figure 3.22: Comparison of PDDP and PILCO in terms of data efficiency and controller learning speed. (a) Number of interactions with the physical systems required to learn the tasks. (b) Total computational time (in minutes) consumed to obtain the final controllers.

terms of data efficiency based on the number of interactions required with the physical systems. For the systems used for testing, PDDP requires around $15\% - 25\%$ more interactions than PILCO. The number of interactions indicates the amount of sampled trajectories required from the physical system to learn the task from scratch. In terms of total computational time required to obtain the final controller, PDDP outperforms PILCO significantly as shown. For the 6 and 12 dimensional systems used for testing, PILCO requires an iterative method (e.g.,CG or BFGS) to solve high dimensional optimization problems (depending on the policy parameterization), while PDDP computes local optimal controls (3.57) without an extra optimizer.



(a) (b) (c) (d)

Figure 3.23: The quadrotor flight task. (a) Quadrotor simulation environment. (b) Expected trajectory cost reduction during PDDP optimization. (c) Trajectory costs over 10 independent trials using the optimized control policy. (d) State trajectories of the quadrotor task collected over 10 independent trials using the optimized controller. Dash lines are desired states.

3.3.2 Constrained Reinforcement Learning

Suppose that the state of the system has to satisfy affine constraints of the form (3.51). We showed that a moment-based approximation with respect to the belief state can be used via (3.53). The question is how one can incorporate such a set of constraints within PDDP. At the level of policy optimization, this is straightforward by using the KKT and Augmented Lagrangian formulations of section 3.1. However, this will not be enough at the level of policy evaluation on the physical system, since we need a way to verify that the applied controls will not violate the constrained (at least with a high probability).

To this end, we will modify the CBF theory to account for the chance constraint. Notice that direct application (2.29) will result in a non-convex optimization problem with respect to the control inputs. To this end, we will proceed as follows:

We begin by enforcing the transition dynamics to be affine in controls (this is a valid assumption for many mechanical/robotics systems). We have

$$\Delta x_i = f_i(\mathbf{x}, \mathbf{u}) + \epsilon,$$

with

$$f_i(x, u) = \phi_{i,0}^{\top}(\mathbf{x}) w_{i,0} + \sum_{j=1}^{m} \phi_{i,j}^{\top}(\mathbf{x}) w_{i,j} u_j,$$

for a given set of basis functions ϕ and weights w. Since the kernel function is constructed as a (possibly infinite-dimensional) dot product between the basis functions, the above form of regressors will correspond to the kernel:

$$k_i(\tilde{\mathbf{x}}, \tilde{\hat{\mathbf{x}}}) = k_{i,0}(\mathbf{x}, \hat{\mathbf{x}}) + \sum_{j=1}^{m} u_j k_{i,j}(\mathbf{x}\hat{\mathbf{x}})$$

which is quadratic in the controls. Direct application of the mean and covariance functions

on a test point $(\mathbf{x}_\star, \mathbf{u}_\star)$ will results in predictions:

$$\begin{aligned}\mu_i(\mathbf{x}_\star, \mathbf{u}_\star) =& \mathbf{u}_\star^\top \beta_i(\mathbf{x}_\star, \mathbf{X}; \boldsymbol{\theta}) \\ \Sigma_i(\mathbf{x}_\star, \mathbf{u}_\star) =& \mathbf{u}_\star^\top \Gamma_i(\mathbf{x}_\star, \mathbf{X}; \boldsymbol{\theta}) \mathbf{u}_\star,\end{aligned}$$

where Γ, β above will be properly defined, and will depend on the state-dependent kernels, training data and hyperparameters.

While we have managed to express the moments as linear/quadratic functions of the controls, we still have to deal with the square root term of (3.53) which imposes a nonlinearity on the norm of controls. To this end, notice that

$$\sqrt{\Sigma_i^{k+1}} = \sqrt{\Sigma_i^k} + \sqrt{\Sigma_i^k}'(\Sigma_i^{k+1} - \Sigma_i^k) + \frac{1}{2}\sqrt{\xi}''(\Sigma_i^{k+1} - \Sigma_i^k)^2,$$

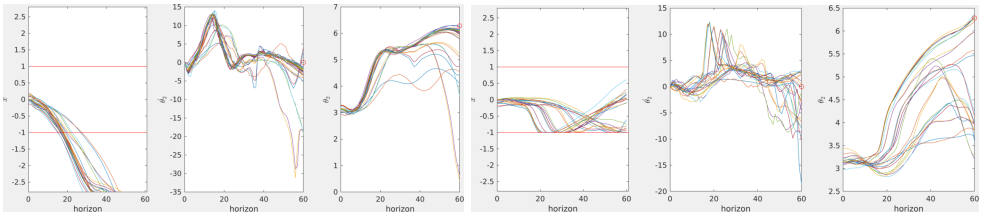
for some $\xi \in \mathbb{R}$ such that $|\xi - \Sigma_i^k| \leq |\Sigma_i^{k+1} - \Sigma_i^k|$. However, $\sqrt{y}'' < 0$ for any $y > 0$ (the origin can be handled via a local polynomial approximation). In this way, given a (noisy) control u_ϵ^k to be used on the actual system for exploration, we can formulate the following sufficient optimization problem

$$\begin{aligned}&\min_{u^k} \quad ||u^k - u_\epsilon^k||^2 \\ &\text{subject to} \\ &\sum_i H_i\big((u^k)^\top \beta_i(x^k, \mathbf{X}; \boldsymbol{\theta}) + \\ &\qquad \text{sign}(H_i)\kappa_\delta\big[\sqrt{\Sigma_i^k} + \sqrt{\Sigma_i^k}'((u^k)^\top \Gamma_i(x^k, \mathbf{X}; \boldsymbol{\theta})u^k - \Sigma_i^k)\big]\big) + d \leq 0 \\ &u_{min} \leq u^k \leq u_{max} \\ &(u^k)^\top \Gamma_i(x^k, \mathbf{X}; \boldsymbol{\theta})u^k \leq \delta,\end{aligned} \tag{3.58}$$

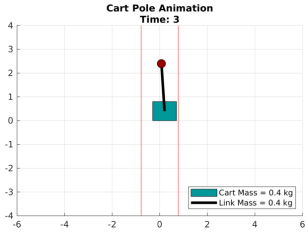
which is a convex QCQP since the Gram matrix is always positive semidefinite and can be solved efficiently in real time. The last constraint is a bound for the variance of the predictions, which we have observed to improve the numerical behavior of the framework.

Table 3.3: Comparison between unconstrained and constrained PDDP

Versions of PDDP	Computation time (minutes)	Total # of system interactions
Unconstrained	30	35
Constrained	40	44



(a) Sampled rollouts: Unconstrained (b) Sampled rollouts: Constrained



(c) Task illustration

Figure 3.24: Sampled rollouts on actual system for each setting of PDDP

Simulations

We evaluate the constrained, model-based RL framework of this section on the inverted double cart pendulum swing up task. Let the state be $\boldsymbol{x} = [x, v, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^\top$, where: x: position of cart, v: velocity of cart, θ_i: ith angle of link, $\dot{\theta}_i$: ith angular velocity of link. We consider the following state constraint: $|x| \geq 1$. Table 3.3 shows the performance comparison between unconstrained and constrained PDDP for the particular task, while Figure 3.24 shows the sampled trajectories on the actual system (only three states are shown, including the position of the cart).

3.4 Geometric Differential Dynamic Programming

In this section we formulate a Lie-theoretic version of DDP for deterministic systems evolving on Lie groups. The derivation goes along the lines of the original DDP method, with each step being modified to account for Lie group formulations of dynamics and cost functions. We thus obtain a numerical, iterative, coordinate-free algorithm for control. We also provide linearization schemes for generic classes of discrete mechanical systems which are required by our method.

We will denote the state and control input of our system at time instant $t = t_k$ by $g^k \in G$ and $u^k \in \mathbb{R}^m$, respectively. The sequence of a state trajectory from t_0 to t_H will be given by $\{g^k\}_0^H = \{g^0, g^1, ..., g^H\}$. For simplicity, we will often omit the indexes and write $\{g^k\}$. Similarly, a control sequence will be denoted by $\{u^k\}_0^{H-1}$.

3.4.1 Derivation

Consider the following discrete-time, finite-horizon optimal control problem:

$$\begin{aligned} &\min_{\{u^k\}_0^{H-1}} \quad J(\{g^k\}_0^H, \{u^k\}_0^{H-1}) \\ \text{s.t.} \quad &g^{k+1} = f^k(g^k, u^k), \quad g^0 = \bar{g}^0 \end{aligned} \tag{3.59}$$

with $J := \sum_{k=0}^{H-1} \mathcal{L}^k(g^k, u^k) + F(g^H)$. Let $\{\bar{u}^k\}_0^{H-1}$ be a nominal control sequence, and let $\{\bar{g}^k\}_0^H$ denote the corresponding nominal state trajectory. We consider perturbations of the nominal control inputs given by $\{u_\epsilon^k\}_0^{H-1} := \{\bar{u}^k + \delta u^k\}_0^{H-1}$. Assuming that $\delta u^k \in \mathbb{R}^m$ is small enough for all k, the perturbed state trajectory, $\{g_\epsilon^k\}_0^H$, will remain close to the nominal one. Therefore, we can use exponential coordinates to write: $\{g_\epsilon^k\} = \{\bar{g}^k \exp(\zeta^k)\}$, with $\zeta^k \in \mathfrak{g}$.

The derivation also requires a linearization scheme for the perturbation vectors, $\{\zeta^k\}$. For now, we will assume that such a scheme is available. For more details about possible

linearization schemes for discrete dynamics on Lie groups, we refer to [41]

$$\zeta^{k+1} \approx$$
$$\Phi^k(\zeta^k) + \mathrm{B}^k(\delta u^k) + \frac{1}{2}\big(\Theta^k(\zeta^k)(\zeta^k) + \Gamma^k(\zeta^k)(\delta u^k) + \Delta^k(\delta u^k)(\zeta^k) + \Xi^k(\delta u^k)(\delta u^k)\big), \tag{3.60}$$

Now, from Bellman's principle of optimality in discrete time, we have

$$V^k(g^k) = \min_{u^k} \big[\mathcal{L}^k(g^k, u^k) + V^{k+1}(g^{k+1})\big]. \tag{3.61}$$

We proceed by expanding both sides of eq. (3.61) about a nominal sequence. The perturbed value function can be written as

$$\begin{aligned} &V^k(\bar{g}^k \exp(\kappa\zeta^k)) = \\ &V^k(\bar{g}^k) + \kappa \left.\frac{\mathrm{d}}{\mathrm{d}s}\right|_{s=0} V^k(\bar{g}^k \exp(s\zeta^k)) + \frac{1}{2}\kappa^2 \left.\frac{\mathrm{d}^2}{\mathrm{d}s^2}\right|_{s=0} V^k(\bar{g}^k \exp(s\zeta^k)) + O(|\kappa|^3) \\ &= V^k(\bar{g}^k) + \kappa \mathsf{D}V^k(\bar{g}^k)(T_e L_{\bar{g}^k}\zeta^k) + \frac{1}{2}\kappa^2 \mathsf{D}(\mathsf{D}V^k(T_e L_{g^k}\zeta^k))(\bar{g}^k)(T_e L_{\bar{g}^k}\zeta^k) + O(|\kappa|^3). \end{aligned} \tag{3.62}$$

The first equality is obtained by treating V^k as a function of $\kappa \in \mathbb{R}$, and the second equality has been proven in [42, eq. 2.12.2] and [23, page 315]. Now, from the definition of the Hessian operator (see section 2.4), the second-order term is equal to $\frac{1}{2}\kappa^2(\mathrm{Hess}V^k(\bar{g}^k)(T_e L_{\bar{g}^k}\zeta^k)(T_e L_{\bar{g}^k}\zeta^k) + T_e L_{\bar{g}^k}\omega(\zeta^k, \zeta^k)(V^k(\bar{g}^k)))$. Assuming G is endowed with the (0), (+), or (-) Cartan connectionn, then the skew-symmetry of $\omega(\cdot,\cdot)$ transforms equation (3.62) into

$$V^k(g_\epsilon^k) =$$
$$V^k(\bar{g}^k) + \mathsf{D}V^k(\bar{g}^k)\big(T_e L_{\bar{g}^k}\zeta^k\big) + \frac{1}{2}\mathrm{Hess}^{(0)}V^k(\bar{g}^k)(T_e L_{\bar{g}^k}\zeta^k)(T_e L_{\bar{g}^k}\zeta^k) + O(||\zeta^k||^3),$$

where we have absorbed κ into ζ^k. Equivalently:

$$V^k(g_\epsilon^k) = V^k(\bar{g}^k) + \mathcal{V}_g^k(\bar{g}^k)\big(\zeta^k\big) + \frac{1}{2}\mathcal{V}_{gg}^k(\bar{g}^k)\big(\zeta^k\big)(\zeta^k\big) + O(||\zeta^k||^3), \tag{3.63}$$

with

$$\mathcal{V}_g^k(g^k) := T_e L_{g^k}^* \circ \mathsf{D}V^k(g^k), \quad \mathcal{V}_{gg}^k(g^k) := T_e L_{g^k}^* \circ \text{Hess}^{(0)} V^k(g^k) \circ T_e L_{g^k}. \tag{3.64}$$

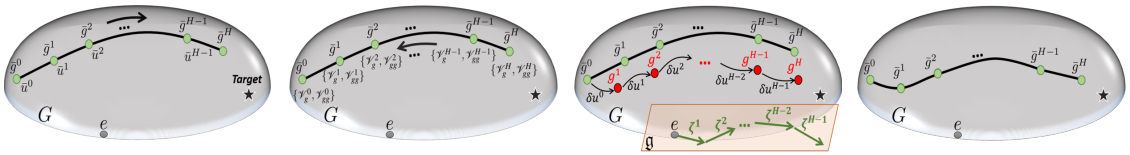
The last manipulation transforms the operators $\mathsf{D}V^k(g^k) : T_{g^k}G \to \mathbb{R}$ and $\text{Hess}^{(0)}V^k(g^k) : T_{g^k}G \to T_{g^k}^*G$, into $\mathcal{V}_g^k(g^k) : \mathfrak{g} \to \mathbb{R}$ and $\mathcal{V}_{gg}^k(g^k) : \mathfrak{g} \to \mathfrak{g}^*$, respectively. In light of this, our algorithm will be derived by solely using operations on $(\mathfrak{g}^*, \mathfrak{g})$. This will allow us, for example, to backpropagate the (trivialized) differential and Hessian of the value function along nominal trajectories. From a computational standpoint, by defining a basis for $\mathfrak{g}$ and its dual, we will be able to implement all steps through standard matrix/vector products.

Using similar arguments and the linearization of ζ, we will expand the *Q function*, $Q^k(g^k, u^k) := \mathcal{L}^k(g^k, u^k) + V^{k+1}(g^{k+1})$, as follows

$$\begin{aligned} &Q^k(g_\epsilon^k, u_\epsilon^k) \approx \\ &Q_0^k + \langle Q_g^k, \zeta^k\rangle + \langle Q_u^k, \delta u^k\rangle + \frac{1}{2}\big(\langle Q_{gg}^k(\zeta^k), \zeta^k\rangle + +2\langle Q_{gu}^k(\zeta^k), \delta u^k\rangle + \langle Q_{uu}^k(\delta u^k), \delta u^k\rangle\big), \end{aligned} \tag{3.65}$$

Now, from the definition of Q function we have $V^k(g_\epsilon^k) = \min_{\delta u^k}\left[Q^k(g_\epsilon^k, u_\epsilon^k)\right]$. Based on the aforementioned approximations and the bilinearity of the natural pairing, the optimal controls are determined as: $\delta u_\star^k = -(Q_{uu}^k)^{-1} \circ Q_u^k - (Q_{uu}^k)^{-1} \circ Q_{gu}^k(\zeta^k)$. As in the finite-dimensional case, we add an external parameter, $\gamma \in (0, 1]$, in the controls update:

$$\delta u_\star^k = -\gamma(Q_{uu}^k)^{-1} \circ Q_u^k - (Q_{uu}^k)^{-1} \circ Q_{gu}^k(\zeta^k), \tag{3.66}$$



(a) (b) (c) (d)

Figure 3.25: Illustration of Differential Dynamic Programming on Lie groups: (a) Given a nominal control sequence, the corresponding trajectory is generated on the configuration manifold, (b) The (trivialized) derivatives of the value function are backpropagated along the nominal trajectory, (c) Control updates are determined that yield a new state and control sequence. This requires computing the linearized state perturbations on the Lie algebra, (d) The updated sequence is treated as the nominal one, and the procedure is repeated until convergence.

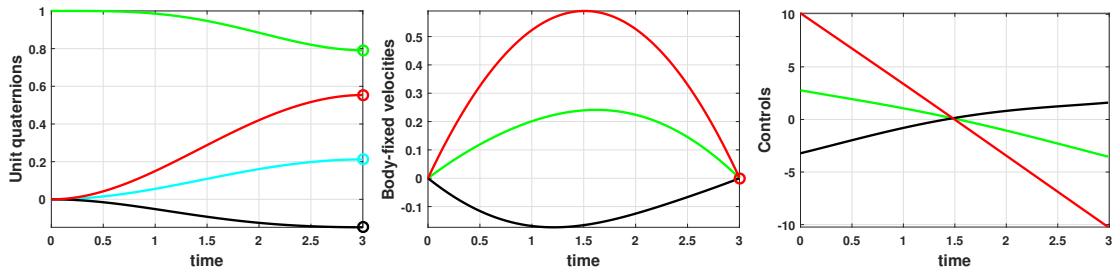


Figure 3.26: Illustration of DDP's (sub)optimal solution - the obtained state trajectory and controls are depicted. The first graph from the left uses unit quaternions to represent the sequence of rotation matrices over time. Each circle at the terminal time instant corresponds to the desired states, R_d or Ω_d.

It remains to backpropagate the trivialized value function and its derivatives to determine (3.66). These expressions will not be included due to space limitations. Figure 3.25 shows an illustration of the applied steps.

The proposed scheme is employed here to control a mechanical system in simulation. We consider the dynamics of a rigid satellite whose state evolves on the tangent bundle $TSO(3)$. The obtained (sub)optimal state trajectory and control sequence are given in Figure 3.26, along with the desired final states, R_d and Ω_d. In these graphs, we have plotted the attitude by using a unit quaternion representation for each rotation matrix.

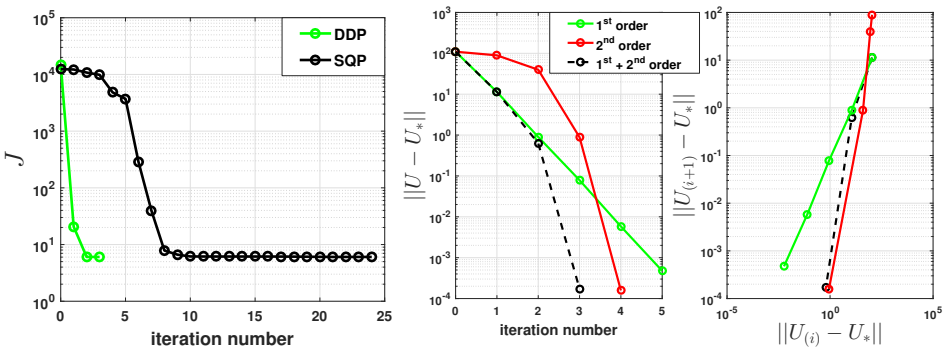


Figure 3.27: Comparison between Differential Dynamic Programming and SQP. The two approaches give the same solution, but DDP takes significantly less time and iterations. Furthermore, the SQP-solver yields infeasible dynamics for a large number of steps. The second and third plots display the convergence of DDP with different linearization schemes.

Simulations

As a benchmark comparison, we also employed MATLAB's built-in SQP implementation. Figure 3.27 compares the total cost per iteration between Differential Dynamic Programming and SQP. The two methods reach the same solution, but DDP requires much fewer iterations. In addition, our MATLAB implementation of DDP converged in 1.9 s, with SQP being approximately 300 times slower. Last but not least, the SQP solver did not yield feasible dynamics until the 9^{th} iteration. This significant difference in performance is observed because a direct optimization method will (i) increase the dimension of the decision vector, (ii) search in a space under many equality constraints, and (iii) scale cubically with the time horizon.

3.4.2 Convergence analysis of geometric DDP

We study the convergence properties of the algorithm developed in this section. An analogous analysis for the Euclidean case can be found in [36] and [37]. Therein, the authors showed that, under some mild conditions, the original DDP method will globally converge to stationary solution. Their work is limited, though, to optimal control problems with terminal costs only. In this section we provide a convergence analysis that deals with the generic problem in (3.59), and handles systems evolving on Lie groups.

In what follows, let $\{\delta u_\star^k\}_0^{H-1}$ be the (sub)optimal control updates given by (3.66). Let also $\bar{U} \in \mathbb{R}^{mH}$ and $\delta U_\star \in \mathbb{R}^{mH}$ denote an entire sequence of nominal inputs and updates, respectively; that is, $\bar{U} := \left((\bar{u}^0)^\top, ..., (\bar{u}^{H-1})^\top\right)^\top$ and $\delta U_\star := \left((\delta u_\star^0)^\top, ..., (\delta u_\star^{H-1})^\top\right)^\top$. The former will be associated with the decision (control) variables at a particular iteration of the DDP algorithm, while the latter will correspond to their updates. We will similarly use $U \in \mathbb{R}^{mH}$ to denote an arbitrary sequence of controls. Moreover, recall from (3.60) that given (small enough) control perturbations $\{\bar{u}^k + \upsilon^k\}$, we can define the perturbed state trajectory as $\{\bar{g}^k \exp(\zeta^k)\}$, with

$$\zeta^{k+1} = \Phi^k(\zeta^k) + \mathrm{B}^k(\upsilon^k) + O(||(\zeta^k, \upsilon^k)||^2), \tag{3.67}$$

for each $k = 0, 1, ..., H-1$. Consider now the following lemma and the resulting theorem.

Lemma 2. *Define $\psi^k \in \mathfrak{g}^*$ as*

$$\begin{aligned} \psi^k &:= \ell_g^k(\bar{g}^k, \bar{u}^k) + (\Phi^k)^* \circ \psi^{k+1}, \quad \text{for } k = 0, ..., H-1, \\ \psi^H &:= T_e L_{\bar{g}^H}^* \circ \mathsf{D}_g F(\bar{g}^H). \end{aligned} \tag{3.68}$$

Then, the total cost of (3.59) *satisfies*

$$\mathsf{D}_{u^k}|_{\bar{U}} J = \ell_u^k(\bar{g}^k, \bar{u}^k) + (\mathrm{B}^k)^* \circ \psi^{k+1}, \tag{3.69}$$

where F is the terminal cost, Φ^k, B^k are determined by the trivialized, linearized dynamics and the "ℓ" functions are the trivialized derivatives of the running cost.

Proof. Let $\{\bar{u}^i + \upsilon^i\}_0^{H-1}$ denote small enough control perturbations about $\bar{U}$. Define also the concatenated vector $\mathsf{U} := ((\upsilon^0)^\top, ..., (\upsilon^{H-1})^\top)^\top \in \mathbb{R}^{m(H-1)}$. Then, it is not hard to see

that

$$\begin{aligned}\mathsf{D}_U|_{\bar{U}} J(\{g^i\},\{u^i\}) &= \mathsf{D}_{\mathsf{U}}|_0 J(\{g_\epsilon^i\},\{u_\epsilon^i\}) \\ &= \mathsf{D}_{\mathsf{U}}|_0 J(\{\bar{g}^i \exp(\zeta^i)\},\{\bar{u}^i+\upsilon^i\}).\end{aligned} \tag{3.70}$$

$\{g_\epsilon^i\}$ denotes here the state trajectory under the perturbed controls $\{u_\epsilon^i\} = \{\bar{u}^i + \upsilon^i\}$ (i.e., $g_\epsilon^{i+1} = f^i(g_\epsilon^i, u_\epsilon^i)$), which can be written using the canonical coordinates. Notice that the last term in (3.70) is only a function of $\{\zeta^i\}$, $\{\upsilon^i\}$. Furthermore, $\{\upsilon^i\}_0^{H-1} \equiv \{0\}_0^{H-1}$ implies $\{\zeta^i\}_0^H \equiv \{0\}_0^H$. Now using the chain rule for mappings on manifolds yields

$$\begin{aligned}&\mathsf{D}_{u^k}|_{\bar{U}} J \overset{(3.70)}{=} \mathsf{D}_{\upsilon^k}|_0 J(\{\bar{g}^i \exp(\zeta^i)\},\{\bar{u}^i+\upsilon^i\}) \\ &\overset{(2.1)}{=} \mathsf{D}_{\upsilon^k}|_0 \Big[\sum_{i=k}^{H-1} \Lambda^i(\bar{g}^i \exp(\zeta^i), \bar{u}^i+\upsilon^i)) + F(\bar{g}^H \exp(\zeta^H))\Big] \\ &= \mathsf{D}_{u^k}\Lambda^k(\bar{g}^k,\bar{u}^k) \\ &\quad + \mathsf{D}_{g^{k+1}}\Lambda^{k+1}(\bar{g}^{k+1},\bar{u}^{k+1}) \circ T_e L_{\bar{g}^{k+1}} \circ \mathsf{D}\exp(0) \circ \left.\frac{\partial \zeta^{k+1}}{\partial \upsilon^k}\right|_0 \\ &\quad + \cdots + \mathsf{D}_{g^H} F(\bar{g}^H) \circ T_e L_{\bar{g}^H} \circ \mathsf{D}\exp(0) \circ \\ &\qquad \left.\frac{\partial \zeta^H}{\partial \zeta^{H-1}} \circ \frac{\partial \zeta^{H-1}}{\partial \zeta^{H-2}} \circ \cdots \circ \frac{\partial \zeta^{k+1}}{\partial \upsilon^k}\right|_0.\end{aligned}$$

Finally, from the identity $\mathsf{D}\exp(0)\cdot\rho = \rho$ and equation (3.67), the above expression

becomes

$$
\begin{aligned}
\mathsf{D}_{u^k}|_{\bar{U}} J =& \ell_u^k + \mathsf{D}_{g^{k+1}} \Lambda^{k+1} \circ T_e L_{\bar{g}^{k+1}} \circ \mathbf{B}^k + \\
& \mathsf{D}_{g^{k+2}} \Lambda^{k+2} \circ T_e L_{\bar{g}^{k+2}} \circ \Phi^{k+1} \circ \mathbf{B}^k + \cdots \\
& + \mathsf{D}_{g^H} F \circ T_e L_{\bar{g}^H} \circ \Phi^{H-1} \circ \Phi^{H-2} \circ \cdots \Phi^{k+1} \circ \mathbf{B}^k \\
\overset{(3.68)}{=} & \ell_u^k + (\mathbf{B}^k)^* \circ \Bigg[\ell_g^{k+1} + (\Phi^{k+1})^* \circ \big[\ell_g^{k+2} + (\Phi^{k+2})^* \circ \big[\cdots \\
& + (\Phi^{H-2})^* \big[\underbrace{\ell_g^{H-1} + (\Phi^{H-1})^* \circ \underbrace{T_e L_{\bar{g}^H}^* \circ \mathsf{D}_{g^H} F}_{\psi^H}}_{\psi^{H-1}} \big] \cdots \big] \Bigg] \\
\overset{(3.68)}{=} & \cdots \\
\overset{(3.68)}{=} & \ell_u^k + (\mathbf{B}^k)^* \circ \psi^{k+1},
\end{aligned}
$$

where we have omitted showing the explicit dependence on $\{\bar{g}^i\}$ and $\{\bar{u}^i\}$ for brevity. □

Theorem 7. *Consider the discrete-time optimal control problem in* (3.59)*, with J being the cost function. Let $\bar{U}$ be a nominal control sequence, and $\delta U_\star$ contain the control updates from* (3.66)*. Then, the following is true:*

$$
\mathsf{D}_U|_{\bar{U}} J \, (\delta U_\star) = -\gamma \sum_{i=0}^{H-1} \langle Q_u^i, (Q_{uu}^i)^{-1} (Q_u^i) \rangle + O(\gamma^2).
$$

Proof. First, we prove the more generic result:

$$
\begin{aligned}
& \sum_{i=k}^{H-1} \mathsf{D}_{u^i}|_{\bar{U}} J \, (\delta u_\star^i) = \\
& - \gamma \sum_{i=k}^{H-1} \langle Q_u^i, (Q_{uu}^i)^{-1} (Q_u^i) \rangle + \langle \mathcal{V}_g^k - \psi^k, \zeta^k \rangle + O(\gamma^2).
\end{aligned}
\tag{3.71}
$$

Let us consider the case when $k \equiv H-1$:

$$\begin{aligned}
\mathsf{D}_{u^{H-1}}|_{\bar{U}} J &\overset{(3.69)}{=} \ell_u^{H-1} + (\mathbf{B}^{H-1})^* \circ \psi^H \\
&= \ell_u^{H-1} + (\mathbf{B}^{H-1})^* \circ T_e L_{\bar{g}^H}^* \circ \mathsf{D}_g F \\
&= \ell_u^{H-1} + (\mathbf{B}^{H-1})^* \circ \mathcal{V}_g^H \\
&= Q_u^{H-1}.
\end{aligned}$$

Moreover, from (3.68)

$$\psi^{H-1} = \ell_g^{H-1} + (\Phi^{H-1})^* \circ \mathcal{V}_g^H = Q_g^{H-1}. \tag{3.72}$$

It is thus clear from (3.66) that

$$\begin{aligned}
&\mathsf{D}_{u^{H-1}}|_{\bar{U}} J\,(\delta u_\star^{H-1}) = \\
&- \gamma \langle Q_u^{H-1}, (Q_{uu}^{H-1})^{-1}(Q_u^{H-1}) + (Q_{uu}^{H-1})^{-1} \circ Q_{ug}^{H-1}(\zeta^{H-1}) \rangle \\
&= -\gamma \langle Q_u^{H-1}, (Q_{uu}^{H-1})^{-1}(Q_u^{H-1}) \rangle - \langle \mathcal{V}_g^{H-1} - Q_g^{H-1}, \zeta^{H-1} \rangle \\
&\overset{(3.72)}{=} -\gamma \langle Q_u^{H-1}, (Q_{uu}^{H-1})^{-1}(Q_u^{H-1}) \rangle - \langle \mathcal{V}_g^{H-1} - \psi^{H-1}, \zeta^{H-1} \rangle.
\end{aligned}$$

Now, assume that identity (3.71) is satisfied for $k+1$. Our goal is to show that it holds for k as well. Note that by using a similar argument as in [36, Lemma 4], one can show that

$\zeta^k = O(\gamma)$ and $\delta u_\star^k = O(\gamma)$, $\forall k$. Then, we obtain

$$
\begin{aligned}
&\sum_{i=k}^{H-1} \mathsf{D}_{u^i}|_{\bar{U}} J\left(\delta u_\star^i\right) \overset{(3.71)}{=} D_{u^k}|_{\bar{U}} J\left(\delta u_\star^k\right) \\
&\quad - \gamma \sum_{i=k+1}^{H-1} \langle Q_u^i, (Q_{uu}^i)^{-1}(Q_u^i)\rangle + \langle \mathcal{V}_g^{k+1} - \psi^{k+1}, \zeta^{k+1}\rangle + O(\gamma^2) \\
&\overset{(3.67),(3.69)}{=\!=} \langle \ell_u^k + (\mathbf{B}^k)^* \circ \psi^{k+1}, \delta u_\star^k\rangle - \gamma \sum_{i=k+1}^{H-1} \langle Q_u^i, (Q_{uu}^i)^{-1}(Q_u^i)\rangle \\
&\qquad + \langle \mathcal{V}_g^{k+1} - \psi^{k+1}, \Phi^k \zeta^k + \mathbf{B}^k \delta u_\star^k\rangle + O(\gamma^2) \\
&= \langle \underbrace{\ell_u^k + (\mathbf{B}^k)^* \circ \mathcal{V}_g^{k+1}}_{Q_u^k}, \delta u_\star^k\rangle - \gamma \sum_{i=k+1}^{H-1} \langle Q_u^i, (Q_{uu}^i)^{-1}(Q_u^i)\rangle \\
&\qquad + \langle (\Phi^k)^* \circ \mathcal{V}_g^{k+1} - \underbrace{(\Phi^k)^* \circ \psi^{k+1}}_{\psi^k - \ell_g^k}, \zeta^k\rangle + O(\gamma^2) \\
&\overset{(3.66)}{=} -\gamma \sum_{i=k}^{H-1} \langle Q_u^i, (Q_{uu}^i)^{-1}(Q_u^i)\rangle + \langle \underbrace{(\Phi^k)^* \circ \mathcal{V}_g^{k+1} + \ell_g^k}_{Q_g^k} - \psi^k, \zeta^k\rangle \\
&\qquad + \langle \underbrace{-Q_{gu}^k \circ (Q_{uu}^k)^{-1} \circ Q_u^k}_{\mathcal{V}_g^k - Q_g^k}, \zeta^k\rangle + O(\gamma^2) \\
&= -\gamma \sum_{i=k}^{H-1} \langle Q_u^i, (Q_{uu}^i)^{-1}(Q_u^i)\rangle + \langle \mathcal{V}_g^k - \psi^k, \zeta^k\rangle + O(\gamma^2).
\end{aligned}
$$

Finally, since the initial state is considered to be fixed, we will have $g^0 = \bar{g}^0 = g_\epsilon^0 = \bar{g}^0 \exp(\zeta^0) = \mathsf{g}^0$, which in turn implies that $\zeta^0 = 0$. Thus, evaluating eq. (3.71) at $k \equiv 0$, gives: $\mathsf{D}_U|_{\bar{U}} J\left(\delta U_\star\right) = \sum_{i=0}^{H-1} \mathsf{D}_{u^i}|_{\bar{U}} J\left(\delta u_\star^i\right) = -\gamma \sum_{i=0}^{H-1} \langle Q_u^i, (Q_{uu}^i)^{-1}(Q_u^i)\rangle + O(\gamma^2)$, which concludes the proof. □

Theorem 7 gives an expression for the directional derivative of the cost function along the control updates $\delta U_\star$ computed by DDP. It implies that when: i) $\gamma > 0$ is small enough, ii) Q_u^i is not zero for all time instances i and iii) Q_{uu}^i is positive definite for all i, DDP will give descent directions for problem (3.59). The following theorem uses this idea to establish the convergence properties of DDP on Lie groups.

Theorem 8. *Let* $U_{(l)} := \left((u^0_{(l)})^\top, ..., (u^{H-1}_{(l)})^\top\right)^\top \in \mathbb{R}^{mH}$ *be the control inputs obtained at the* l*th iteration of DDP, with* $l = 0, 1, ...$ *. Assume that*

i) Q^k_{uu} *is positive definite for all time instances* $k = 0, ..., H-1$ *and iterations* $l = 0, 1, ...$ *.*

ii) The level set $\mathcal{L}_0 := \{U \in \mathbb{R}^{mH} : J(\{g^k\}^H_0, \{u^k\}^{H-1}_0) \leq J(\{g^k_{(0)}\}^H_0, \{u^k_{(0)}\}^{H-1}_0)\}$ *is compact. Here* $\{g^k_{(0)}\}$ *corresponds to the state trajectory under controls* $U_{(0)}$*, determined by the dynamics equations in* (3.59)*.*

Then

i) A control trajectory $\breve{U} \in \mathbb{R}^{mH}$ *is a stationary point of* J *if* $Q^k_u(\breve{U}) \equiv 0$ *for each* k*.*

ii) Given a nominal set of controls $\bar{U}$ *for which* $Q^k_u(\bar{U}) \equiv 0$*,* $\forall k$*, the control updates of DDP will satisfy* $\delta u^k_\star = 0$*,* $\forall k$*.*

iii) The sequence of controls $\{U_{(l)}\}$ *lies in* $\mathcal{L}_0$*, and any subsequence of* $\{U_{(l)}\}$ *has an accumulation point.*

iv) The sequence of costs $\{J(\{g^k_{(l)}\}^H_0, \{u^k_{(l)}\}^{H-1}_0)\}$ *with index "*l*" is non-increasing, and there exists* $\breve{J} \in \mathbb{R}$ *such that* $\lim_{l\to\infty} J(\{g^k_{(l)}\}^H_0, \{u^k_{(l)}\}^{H-1}_0) = \breve{J}$*.*

v) Any accumulation point of the sequence of controls $\{U_{(l)}\}$ *is a stationary point of* J*.*

vi) When the number of stationary points of J *is finite, the sequence* $\{U_{(l)}\}$ *converges to a unique stationary solution of* (3.59)*.*

Proof. $i)$ For this part of the proof it will be implied that the linearized dynamics in (3.60), as well as the mappings Q^k, V^k, Λ^k and their (trivialized) derivatives, are evaluated on the given control trajectory $\breve{U}$ (in other words, we will simply treat $\breve{U}$ as a nominal control trajectory within DDP).

Assume that $Q_u^k = 0$ for each k. Then

$$\mathcal{V}_g^k = Q_g^k - Q_{gu}^k \circ (Q_{uu}^k)^{-1} \circ Q_u^k = Q_g^k, \quad \forall k. \tag{3.73}$$

Based on this, we get from (3.68)

$$\begin{aligned}
\psi^{H-1} &= \ell_g^{H-1} + (\Phi^{H-1})^* \circ \mathcal{V}_g^H = Q_g^{H-1} \overset{(3.73)}{=} \mathcal{V}_g^{H-1}, \\
\psi^{H-2} &= \ell_g^{H-2} + (\Phi^{H-1})^* \circ \psi^{H-1} = \ell_g^{H-2} + (\Phi^{H-1})^* \circ \mathcal{V}_g^{H-1} \\
&= Q_g^{H-2} \overset{(3.73)}{=} \mathcal{V}_g^{H-2}, \\
&\vdots \\
\psi^k &= \mathcal{V}_g^k, \quad \forall k.
\end{aligned}$$

Hence, (3.69) reads: $\mathsf{D}_{u^k}|_{\breve{U}} J = \ell_u^k + (\mathrm{B}^k)^* \circ \mathcal{V}_g^{k+1} \overset{(??)}{=} Q_u^k = 0$, which means that $\breve{U}$ is a stationary point for the cost function.

ii) Assume that $Q_u^k = 0$ for each k, given a particular nominal control trajectory $\bar{U}$ within DDP. Then, recalling that $\zeta^0 = 0$, we obtain from the control update (3.66): $\delta u_\star^0 = -(Q_{uu}^0)^{-1}(Q_u^0) = 0$. From eq. (3.60), this in turn yields $\zeta^1 = 0$. Next, assume that $\delta u_\star^k = 0$ and $\zeta^{k+1} = 0$, for some $k > 0$. Then, $\delta u_\star^{k+1} = -(Q_{uu}^{k+1})^{-1}(Q_u^{k+1}) - (Q_{uu}^{k+1})^{-1} \circ Q_{ug}^{k+1}(\zeta^{k+1}) = 0$. By induction, this proves that $\delta u_\star^k = 0$ for all k.

iii) For the remaining part of the proof, we will denote the sequence of costs by $\{\tilde{J}(U_{(l)})\}$ instead of the more cumbersome notation $\{J(\{g_{(l)}^k\}_0^H, \{u_{(l)}^k\}_0^{H-1})\}$. We recall that $\tilde{J} : \mathbb{R}^{mH} \to \mathbb{R}$ corresponds to the cost function J, composed with the transition dynamics f^k (given in (3.59)). Since a state trajectory can be fully determined by the corresponding control trajectory, the two notations are equivalent.

Let us also define for brevity the term

$$\Upsilon(U_{(l)}) := \sum_{i=0}^{H-1} \langle Q_u^i(U_{(l)}), (Q_{uu}^i(U_{(l)}))^{-1}(Q_u^i(U_{(l)})) \rangle, \tag{3.74}$$

where the notation $Q^i_{uu}(U_{(l)}), Q^i_u(U_{(l)})$ means that the aforementioned operators are evaluated on $U_{(l)}$. Furthermore, recall that at the lth iteration of DDP, $U_{(l)}$ corresponds to the nominal trajectory of the algorithm, and $U_{(l+1)}$ to the updated one. That is,

$$U_{(l+1)} = U_{(l)} + \delta U_{(l)}, \tag{3.75}$$

where $\delta U_{(l)}$ is equivalent to DDP's control perturbations $\delta U_\star := \left((\delta u^0_\star)^\top, ..., (\delta u^{H-1}_\star)^\top\right)^\top$ at its lth iteration, and $\delta u^k_\star$ has been defined in (3.66). Let us now use (3.74) to rewrite Theorem 7 as

$$\mathsf{D}_U|_{U_{(l)}} J\left(\delta U_{(l)}\right) = -\gamma\, \Upsilon(U_{(l)}) + O(\gamma^2). \tag{3.76}$$

We will now focus on proving statement 3. Let us initially assume that $U_{(1)}, ..., U_{(l)} \in \mathcal{L}_0$ and $Q^i_u(U_{(l)}) = 0$ for all i and some iteration l of the algorithm. Then, part 2 of Theorem 8 states that $\delta U_{(l)} = 0$. Hence, statement 3 follows.

Suppose in contrast that Q^i_u is not zero for some instance i within iteration l. From eq. (3.74) and assumption 1, we then have that $\Upsilon(U_{(l)}) > 0$. Since $\delta u^k_{(l)} = O(\gamma)$, equation (3.76) implies that there exists $\gamma_\star \in (0, 1]$ such that

$$\tilde{J}(U_{(l+1)}) - \tilde{J}(U_{(l)}) \leq -\gamma\, \Upsilon(U_{(l)}), \quad \forall \gamma \in (0, \gamma_\star]. \tag{3.77}$$

In other words, the sequence of costs $\{\tilde{J}(U_{(l)})\}$ will be monotonically decreasing and, therefore, by the definition of $\mathcal{L}_0$ we will have that $\{U_{(l)}\} \in \mathcal{L}_0$. Since $\mathcal{L}_0$ is compact by assumption 2, the sequence $\{U_{(l)}\}$ will be bounded, and thus any subsequence $\{U_{(l_j)}\}$ will have an accumulation point in $\mathcal{L}_0$ [43, section 2.3.2]. This completes the proof for 3.

iv) We have already shown that $\{\tilde{J}(U_{(l)})\}$ is a non-increasing sequence from equation (3.77) and the discussion in part 3 of the proof. We also showed in part 3 that $\{U_{(l)}\} \in \mathcal{L}_0$, while $\mathcal{L}_0$ is assumed to be compact from 2. Since J is continuous by assumption

(differentiability of the cost has been assumed in the derivation of DDP), it will have a global minimum in $\mathcal{L}_0$, and thus the non-increasing sequence $\{\tilde{J}(U_{(l)})\}$ will be bounded below. It follows that $\{\tilde{J}(U_{(l)})\}$ is convergent [10, appendix A].

$v)$ Let $\breve{U} \in \mathbb{R}^{mH}$ be any accumulation point of $\{U_{(l)}\}$, associated with a subsequence $\{U_{(l_j)}\}$. From equation (3.77) and parts 3 $\&$ 4 of the proof, we can deduce that

$$\lim_{j\to\infty} \Upsilon(U_{(l_j)}) = \Upsilon(\breve{U}) = 0.$$

From assumption 1, this further means that $Q_u^k = 0$ for all k, when evaluated on the control trajectory $\breve{U}$. Finally, part 1 of Theorem 8 implies that $\breve{U}$ will be a stationary point for J, which gives the desired result.

$vi)$ Convergence to a set of stationary solutions has been shown in part 5. It remains to show that the sequence of control inputs $\{U_{(l)}\}$ will be convergent. From parts 1 - 5 of the proof, we can deduce that $\Upsilon(U_{(l)}) \to 0$ and $\delta U_{(l)} \to 0$. This result, along with the update equation (3.75), implies that $\lim_{l\to\infty} ||U_{(l+1)} - U_{(l)}|| = 0$. Since the number of stationary points of J is assumed to be finite, the latter fact is a sufficient condition for the existence of a unique accumulation point [43, section 2.3.3], which means that $\{U_{(l)}\}$ is convergent. This concludes the proof. □

Remark 2. *Notice that $T\mathbb{R}^m \simeq \mathbb{R}^m$ can simply be viewed as the set of all column vectors in $\mathbb{R}^m$, while its dual will comprise of row vectors in $\mathbb{R}^{1\times m}$. Under this identification, we can account for assumption 1 of Theorem 8 by setting*

$$Q_{uu}^k \leftarrow Q_{uu}^k + \lambda^k I_m,$$

at each k. Here, $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix, and $\lambda^k > 0$ is a regularization parameter that enforces the positive definiteness of Q_{uu}^k for all k [10]. Although there are many ways to find a suitable λ^k, we used a method similar to Levenberg-Marquardt [10] in our simulations: We set $\lambda^k = |\lambda_{uu,\min}^k| + \bar{\lambda}^k$, where $\lambda_{uu,\min}^k$ denotes the most negative

eigenvalue of Q_{uu}^k (if all eigenvalues are nonnegative, this term is 0) and $\bar{\lambda}^k$ is a positive constant that is selected adaptively. Specifically, when $\delta U_\star$ improves the cost function, $\bar{\lambda}^k$ is reduced for the next iteration. Conversely, we increase $\bar{\lambda}^k$ and recompute $\delta U_\star$ (thus, relying less on Q_{uu}^k).

Finally, assumption 2 is required to prove existence of accumulation points. Such assumptions are usually considered for certain classes of optimization methods in order to establish convergence properties (see, e.g., [43, Section 2.3.2]).

CHAPTER 4
SAMPLING-BASED METHODS FOR FINITE- AND INFINITE-DIMENSIONAL SYSTEMS

So far we have studied nonlinear control algorithms which utilize first- and/or second-order expansions of the cost and the dynamics. Gradient information allows the above methods to attain fast convergence. Nonetheless, in many practical applications gradient information may not be available, and thus a different optimization strategy needs to be considered. These include problems with inherent discontinuities, such as contact dynamics [44], or environments where only evaluations of the costs and dynamics are provided [45]. Potentially, one can use smoothing techniques and/or learn data-based models to provide gradient information, however numerical instabilities and poor generalizations may often arise.

Due to the aforementioned limitations, sampling-based techniques have been extensively used over the past few years. Even though convergence is typically slower than their gradient-based counterparts, their ability to handle discontinuities and lack of gradient information is highly desirable. Popular implementations include path integral control [16] and the cross entropy method [46], which have recently found application on real robotics settings [47]. Loosely speaking, these schemes compute control updates based on the evaluation of the costs associated with sampled trajectories. In this direction, this chapter will focused on sampling-based techniques for control for finite- and infinite-dimensional systems.

4.1 Information theoretic control for stochastic systems

In this section, we will address an information theoretic framework for control of stochastic partial differential equations (SPDEs). Although we will focus on dynamics described as stochastic fields (or infinite-dimensional objects) our derivation can be seen as a general-

ization of the finite-dimensional case (i.e., for SDEs) [17].

4.1.1 Variational optimization for parameterized control policies

Let the uncontrolled and controlled version of an H-valued process be given respectively by:

$$\mathrm{d}X = \big(\mathcal{A}X + F(t,X)\big)\mathrm{d}t + \frac{1}{\sqrt{\rho}}G(t,X)\mathrm{d}W(t), \tag{4.1}$$

$$\mathrm{d}\tilde{X} = \big(\mathcal{A}\tilde{X} + F(t,\tilde{X})\big)\mathrm{d}t + G(t,\tilde{X})\big(\mathcal{U}(t,\tilde{X})\mathrm{d}t + \frac{1}{\sqrt{\rho}}\mathrm{d}W(t)\big), \tag{4.2}$$

both with initial condition: $X(0) = \tilde{X}(0) = \xi$. Here, $W \in U$ is a cylindrical Wiener process on $(\Omega, \mathcal{F}, \mathbb{P})$. Following the reasoning of Theorem 4 and its proof, the uncontrolled dynamics from (4.1) can be equivalently written as:

$$\mathrm{d}X = \big(\mathcal{A}X + F(t,X)\big)\mathrm{d}t + G(t,X)\big(\mathcal{U}(t,X)\mathrm{d}t + \frac{1}{\sqrt{\rho}}\mathrm{d}\hat{W}(t)\big), \tag{4.3}$$

on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Here, $\hat{W}$ denotes a cylindrical Wiener process with respect to another measure $\mathbb{Q}$. The law of the uncontrolled states, $\mathcal{L}(\cdot)$, defines a measure on the path space via (4.1) as $\mathcal{L}(\Gamma) := \mathbb{P}(\omega | X(\cdot,\omega) \in \Gamma)$. Similarly, the law of controlled trajectories is $\tilde{\mathcal{L}}(\Gamma) := \mathbb{P}(\omega | \tilde{X}(\cdot,\omega) \in \Gamma)$. We recall that the mapping $X : \Omega \to C([0,T];H) := C$ can be treated as a random variable with values on the Banach space of continuous functions.

We will derive controllers by formulating an optimization problem that utilizes the measure theoretic approach. We suppose that there exists an optimal controller $\mathcal{U}^*$ which corresponds to the law of optimal trajectories, $\mathcal{L}^*(\cdot)$. The optimality of the controller $\mathcal{U}^*$ is with respect to the thermodynamic inequality in (4.30). Furthermore, this controller is given in an implicit fashion, through the optimal measure formulation in (4.31). To compute a control law in an explicit and implementable fashion, we are looking for a control input

$\mathcal{U}(\cdot)$ that minimizes the distance to the optimal path law. That is:

$$\mathcal{U}^*(\cdot) = \underset{\mathcal{U}(\cdot)}{\operatorname{argmax}}\, S(\mathcal{L}^*||\tilde{\mathcal{L}}). \tag{4.4}$$

Under the parameterization $\mathcal{U}(t,X) = \mathcal{U}(t,X;\boldsymbol{\theta})$ the problem above will take the form:

$$\begin{aligned}\boldsymbol{\theta}^* &= \operatorname{argmax}\left[-\int_C \frac{\mathrm{d}\mathcal{L}^*(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})}\log_e\frac{\mathrm{d}\mathcal{L}^*(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})}\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})\right]\\ &= \operatorname{argmin}\left[\int_C \log_e\frac{\mathrm{d}\mathcal{L}^*(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})}\mathrm{d}\mathcal{L}^*(\mathsf{x})\right],\end{aligned} \tag{4.5}$$

where θ is a finite-dimensional set of control parameters. To optimize, we will consider the chain rule property for the Radon-Nikodym derivative. For instance, this results in the following expression:

$$\frac{\mathrm{d}\mathcal{L}^*(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})} = \frac{\mathrm{d}\mathcal{L}^*(\mathsf{x})}{\mathrm{d}\mathcal{L}(\mathsf{x})}\frac{\mathrm{d}\mathcal{L}(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})}. \tag{4.6}$$

Note that the first derivative is given by (4.31) while the second derivative is given by the change of measure between control and uncontrolled infinite dimensional stochastic dynamics. Based on the discussion of Theorem 4, $\tilde{\mathcal{L}}(\Gamma) = \mathbb{Q}(\omega|X(\cdot,\omega)\in\Gamma)$ and $\mathcal{L}^*(\Gamma) = \mathbb{Q}^*(\omega|X(\cdot,\omega)\in\Gamma)$, where $\mathbb{Q}^*$ is properly defined. Assuming the technical assumptions of Theorem 4 are satisfied, (2.22) implies[1]

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\tilde{\mathcal{L}}} = \exp\left(-\int_0^T \langle\psi(s),\mathrm{d}W(s)\rangle_U + \frac{1}{2}\int_0^T ||\psi(s)||_U^2\mathrm{d}s\right), \tag{4.7}$$

where in our case, $\psi(t) := \sqrt{\rho}\mathcal{U}(t,X(t);\boldsymbol{\theta}) \in U$.

For now, we will consider open loop controllers and will parameterize $\mathcal{U}$ as:

$$\mathcal{U}(t)(\mathbf{x}) = \sum_{\ell=1}^{N} m_\ell(\mathbf{x})u_\ell(t) = \mathbf{m}(\mathbf{x})^\top\mathbf{u}(t;\boldsymbol{\theta}). \tag{4.8}$$

Here, $\mathbf{x}\in\mathbb{R}^n$ denotes the spatial component of the SPDEs and $m_\ell \in U$ are design functions

[1]We also use the fact that for any function ψ on C, we have $\mathbb{E}[\psi(X)] = \int_\Omega \psi\big(X(\cdot,\omega)\big)\mathrm{d}\mathbb{P}(\omega) = \int_C \psi(\mathsf{x})\mathrm{d}\mathcal{L}(\mathsf{x})$, see [48, Chapter 1].

that specify how actuation is incorporated into the infinite dimensional dynamical system.

The parameterization in (4.8) can be used for both open loop trajectory optimization as well as for model predictive control. In our simulations we will apply model predictive control through re-optimization, and turn (4.8) into an implicit feedback. Optimization using (4.5) of feedback policies, namely policies that explicitly depend on the stochastic field, is also possible but it is not included in this work. Deriving the optimization scheme for these policies requires a slight modification of the mathematical analysis in this work.

Under the parameterization above, the change of measure between the two SPDEs takes the form:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\tilde{\mathcal{L}}} = \exp\left(-\sqrt{\rho}\int_0^T \mathbf{u}(t)^\top \bar{\mathbf{m}}(t) + \rho\frac{1}{2}\int_0^T \mathbf{u}(t)^\top \mathbf{M}\mathbf{u}(t)\mathrm{d}t\right), \tag{4.9}$$

where

$$\bar{\mathbf{m}}(t) := \left[\langle m_1, \mathrm{d}W(t)\rangle_{U_0}, ..., \langle m_N, \mathrm{d}W(t)\rangle_{U_0}\right]^\top \in \mathbb{R}^N, \tag{4.10}$$

$$\mathbf{M} \in \mathbb{R}^{N\times N}, \quad (\mathbf{M})_{ij} := \langle m_i, m_j\rangle_U. \tag{4.11}$$

To simplify the optimization in (4.5), we will further parameterize $u(t;\theta)$ as a simple (measurable) function:

$$\mathbf{u}(t) = \mathbf{u}_i \in \mathbb{R}^N, \quad \text{if} \quad i\Delta t \le t < (i+1)\Delta t, \tag{4.12}$$

with $i = \{0, 1, \dots L\}$ and $t \in [0, T]$. In this case, the parameters $\boldsymbol{\theta}$ will consist of all step functions $\{\mathbf{u}_i\}$. The following lemma provides the optimal control parameters under the aforementioned representations.

Lemma 3. *(Variational Stochastic Control) For the controlled SPDE in* (4.2)*, consider the*

following objective function:

$$\mathbf{u}^* = \operatorname{argmax} S(\mathcal{L}^* || \tilde{\mathcal{L}}). \tag{4.13}$$

The probability measure $\mathcal{L}^*$ is induced by the optimal controlled SPDE (4.2)*, is optimal with respect to the thermodynamic inequality* (4.30)*, and has the form:*

$$\mathrm{d}\mathcal{L}^*(\mathsf{x}) = \frac{\exp\big(-\rho\mathcal{J}(\mathcal{X})\big)\mathrm{d}\mathcal{L}(\mathsf{x})}{\int_{\mathcal{C}} \exp\big(-\rho\mathcal{J}(\mathcal{X})\big)\mathrm{d}\mathcal{L}(\mathsf{x})}, \tag{4.14}$$

The probability measure $\tilde{\mathcal{L}}$ will be induced by controlled trajectories of the SPDEs under some parameterized policy $\mathcal{U}(t, X; \boldsymbol{\theta})$. When the representations in (4.8) *and* (4.12) *are used, the optimal control parameters are given by:*

$$\begin{aligned} \mathbf{u}_i^* &= \frac{1}{\sqrt{\rho}\Delta t}\mathbf{M}^{-1}\mathbb{E}_{\mathcal{L}}\left[\frac{\exp(-\rho J)}{\mathbb{E}_{\mathcal{L}}[\exp(-\rho J)]}\delta\mathbf{u}_i\right], \quad \textit{and} \\ \delta\mathbf{u}_i &:= \int_{t_i}^{t_{i+1}} \bar{\mathbf{m}}(t). \end{aligned} \tag{4.15}$$

Proof. Under the parameterization $\mathcal{U}(\mathbf{x}, t) = \mathbf{m}(\mathbf{x})^{\mathrm{T}}\mathbf{u}(t)$, the problem takes the form:

$$\mathbf{u}^* = \operatorname{argmin}\left[\int_C \log_e \frac{\mathrm{d}\mathcal{L}^*(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})}\mathrm{d}\mathcal{L}^*(\mathsf{x})\right] = \operatorname{argmin}\left[\int_{\mathcal{C}} \log_{\urcorner} \frac{\mathrm{d}\mathcal{L}^*(\mathsf{x})}{\mathrm{d}\mathcal{L}(\mathsf{x})}\frac{\mathrm{d}\mathcal{L}(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})}\mathrm{d}\mathcal{L}^*(\mathsf{x})\right].$$

Minimization of the last expression is equivalent to the minimization of the expression:

$$\mathbb{E}_{\mathcal{L}^*}\left[\log_e \frac{\mathrm{d}\mathcal{L}(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})}\right] = -\sqrt{\rho}\mathbb{E}_{\mathcal{L}^*}\left[\int_0^T \mathbf{u}(t)^\top \bar{\mathbf{m}}(t)\right] + \frac{1}{2}\rho\mathbb{E}_{\mathcal{L}^*}\left[\int_0^T \mathbf{u}(t)^\top \mathbf{M}\mathbf{u}(t)\mathrm{d}t\right].$$

Since we apply the control in discrete time instances, it suffices to consider the class of step functions:

$$\mathbb{E}_{\mathcal{L}^*}\left[\log_e \frac{\mathrm{d}\mathcal{L}(\mathsf{x})}{\mathrm{d}\tilde{\mathcal{L}}(\mathsf{x})}\right] = -\sqrt{\rho}\sum_{i=0}^{L-1} \mathbf{u}_i^\top \mathbb{E}_{\mathcal{L}^*}\left[\int_{t_i}^{t_{i+1}} \bar{\mathbf{m}}(t)\right] + \frac{1}{2}\rho\sum_{i=0}^{L-1} \mathbf{u}_i^\top \mathbf{M}\mathbf{u}_i\Delta t,$$

where we have used the fact that $\mathbf{M}$ is constant with respect to time. Due to the symmetry of $\mathbf{M}$, minimization of the expression above with respect to $\mathbf{u}_i$ results in:

$$\mathbf{u}_i^* = \frac{1}{\sqrt{\rho}\Delta t}\mathbf{M}^{-1}\mathbb{E}_{\mathcal{L}^*}\left[\int_{t_i}^{t_{i+1}} \bar{\mathbf{m}}(t)\right]. \tag{4.16}$$

Since we cannot sample according to $\mathcal{L}^*$ directly, we need to express the above expectation with respect to the measure induced by uncontrolled dynamics, $\mathcal{L}$. We can then directly sample uncontrolled trajectories based on $\mathcal{L}$ and approximate the optimal controls. The change in expectation is achieved by applying the Radon-Nikodym derivative. The result is equation (4.15). □

4.1.2 Iterative Control of SPDEs

Equation (4.15) requires computation of the expected state cost under the uncontrolled SPDE. For generic problems, we will have to resort to Monte Carlo estimates. As in the finite-dimensional case though, such estimates can rarely be estimated efficiently due to the difficulty of sampling the entire state space. To this end, we will derive an iterative scheme via importance sampling between successively obtained controllers.

By relying on arguments similar to those used in proving Theorem 4, we will develop our scheme on abstract Hilbert spaces, avoiding thus dependencies upon specific model-reduction methods. The obtained methodology can be implemented in a receding horizon fashion and will be used in section 4.1.4 to control various stochastic fields.

Let us denote the controlled dynamics at iteration i by:

$$\mathrm{d}X^{(i)} = \big(\mathcal{A}X^{(i)} + F(t, X^{(i)})\big)\mathrm{d}t + G(t, X^{(i)})\big(\mathcal{U}^{(i)}\mathrm{d}t + \frac{1}{\sqrt{\rho}}\mathrm{d}W(t)\big), \tag{4.17}$$

where $\mathcal{U}^i(t)$ denotes the control at the i^{th} iteration. Using an approach similar to the proof

of Theorem 4, we can express the uncontrolled dynamics in the equivalent form:

$$\begin{aligned} \mathrm{d}X &= \big(\mathcal{A}X + F(t,X)\big)\mathrm{d}t + \frac{1}{\sqrt{\rho}}G(t,X)\mathrm{d}W(t) \\ &= \big(\mathcal{A}X + F(t,X)\big)\mathrm{d}t + G(t,X)\big(\mathcal{U}^{(i)}\mathrm{d}t + \frac{1}{\sqrt{\rho}}\mathrm{d}W^{(i)}(t)\big), \end{aligned} \tag{4.18}$$

where $W^{(i)}$ is a cylindrical Wiener process with respect to some measure $\mathbb{Q}^{(i)}$ with:

$$W^{(i)}(t) := W(t) - \int_0^t \sqrt{\rho}\mathcal{U}^{(i)}(s)\mathrm{d}s. \tag{4.19}$$

Again, we define here the path measure $\mathcal{L}^i(\Gamma) := \mathbb{P}(\omega | X^{(i)}(\cdot,\omega) \in \Gamma)$ induced by (4.17) and the measure $\mathcal{L}(\Gamma) := \mathbb{P}(\omega | X(\cdot,\omega) \in \Gamma)$ induced by (4.18). One can then show the following:

Lemma 4. *(Iterative Stochastic Control) Consider the controlled SPDE in* (4.2) *and a parameterization of the control as specified by* (4.8) *and* (4.12)*. The iterative control scheme for solving the stochastic control problem as formulated in* (4.13) *is given by the following expression:*

$$\mathbf{u}_j^{*(i+1)} = \mathbf{u}_j^{*(i)} + \frac{1}{\sqrt{\rho}\Delta t}\mathbf{M}^{-1}\mathbb{E}_{\mathcal{L}^{()}}\left[\frac{\exp(-\rho J^{(i)})}{\mathbb{E}_{\mathcal{L}^{()}}[\exp(-\rho J^{(i)}]}\delta\mathbf{u}_j^{(i)}\right], \tag{4.20}$$

where we have defined: $J^{(i)} := J + \zeta^{(i)}$ *and*

$$\delta\mathbf{u}_j^{(i)} := \int_{t_j}^{t_{j+1}} \bar{\mathbf{m}}^{(i)}(t), \tag{4.21}$$

$$\zeta^{(i)} := \frac{1}{\sqrt{\rho}}\sum_{k=0}^{L-1}\mathbf{u}_k^{(i)\top}\int_{t_k}^{t_{k+1}}\bar{\mathbf{m}}^{(i)}(t) + \frac{1}{2}\sum_{k=0}^{L-1}\mathbf{u}_k^{(i)\top}\mathbf{M}\mathbf{u}_k^{(i)}\Delta t, \tag{4.22}$$

$$\bar{\mathbf{m}}^{(i)}(t) := \Big[\langle m_1, \mathrm{d}W^{(i)}(t)\rangle_U, ..., \langle m_N, \mathrm{d}W^{(i)}(t)\rangle_U\Big]^\top \in \mathbb{R}^N. \tag{4.23}$$

The expectation in (4.20) *is taken with respect to the probability path measure* $\mathcal{L}^{(i)}$ *,*

which is induced by the controlled SPDE in (4.17).

Proof. We have:

$$\frac{\mathrm{d}\mathcal{L}^{(i)}}{\mathrm{d}\mathcal{L}} = \exp\big(\sqrt{\rho}\sum_{k=0}^{L-1}\mathbf{u}_k^{(i)\top}\int_{t_k}^{t_{k+1}}\bar{\mathbf{m}}^{(i)}(t) + \rho\frac{1}{2}\sum_{k=0}^{L-1}\mathbf{u}_k^{(i)\top}\mathbf{M}\mathbf{u}_k^{(i)}\Delta t\big), \tag{4.24}$$

In order to derive the iterative scheme, we perform one step of importance sampling and express the associated expectations with respect the measure induced by the controlled SPDE in (4.17). Let us begin by modifying (4.15) via the appropriate change of measure:

$$\mathbf{u}_j^{(i+1)} = \frac{1}{\sqrt{\rho}\Delta t}\mathbf{M}^{-1}\mathbb{E}_{\mathcal{L}^{()}}\left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\mathcal{L}^{()}}\frac{\exp(-\rho J)}{\mathbb{E}_{\mathcal{L}}[\exp(-\rho J)]}\delta\mathbf{u}_j\right], \tag{4.25}$$

Regarding $\delta\mathbf{u}_j$, one has:

$$\begin{aligned}\left(\int_{t_j}^{t_{j+1}}\bar{\mathbf{m}}(t)\right)_l &= \int_{t_j}^{t_{j+1}}\langle m_l, \mathrm{d}W(t)\rangle_{U_0} = \int_{t_j}^{t_{j+1}}\langle m_l, \mathrm{d}W^{(i)}(t) + \sqrt{\rho}\mathcal{U}^{(i)}(t)\mathrm{d}t\rangle_{U_0}\\ &= \int_{t_j}^{t_{j+1}}\langle m_l, \mathrm{d}W^{(i)}(t)\rangle_{U_0} + \sqrt{\rho}\Big[\langle m_l, m_1\rangle_{U_0}, ..., \langle m_l, m_N\rangle_{U_0}\Big]\mathbf{u}_j^{(i)}\Delta t.\end{aligned}$$

It follows that:

$$\int_{t_j}^{t_{j+1}}\bar{\mathbf{m}}(t) = \int_{t_j}^{t_{j+1}}\bar{\mathbf{m}}^{(i)}(t) + \sqrt{\rho}\Delta t\mathbf{M}\mathbf{u}_j^{(i)}.$$

Substitution of the Radon-Nikodym derivative yields the final result in (4.20). □

Note that $\mathbb{Q}^{(i)}$ renders $W^{(i)}$ a standard cylindrical Wiener process. Hence, for implementation we will approximate $\delta\mathbf{u}^{(i)}$ by:

$$(\delta\tilde{\mathbf{u}}_j^{(i)})_l := \sum_{s=1}^{R}\langle m_l, \sqrt{\lambda}_s e_s\rangle_U \Delta\beta_s^{(i)}(t_j), \tag{4.26}$$

where $\Delta\beta_s^{(i)}(t_j) \sim \mathcal{N}(0, \Delta t)$ under $\mathbb{Q}^{(i)}$.

4.1.3 Boundary Control of Stochastic Parabolic Equations

We will now examine stochastic optimal control problems subject to boundary control and noise. The key point lies in writing such dynamical systems via the abstract semilinear formulation of (2.19). As shown in [49] and references therein, this can be readily accomplished for a generic class of parabolic systems. Specifically, we consider systems that evolve under the *mild* form:

$$\begin{aligned} X(t) &= e^{t\mathcal{A}}\xi + \int_0^t e^{(t-s)\mathcal{A}}F_1(t,X)\mathrm{d}s \\ &+ (\lambda I - \mathcal{A})\Big[\int_0^t e^{(t-s)\mathcal{A}}\mathsf{D}\big(F_2(t,X) + G(t,X)U(t,X)\big)\mathrm{d}s \\ &\qquad + \int_0^t e^{(t-s)\mathcal{A}}\mathsf{D}B(t,X)\mathrm{d}V(s)\Big], \quad \mathbb{P}\,a.s. \end{aligned} \tag{4.27}$$

Here, the operator D corresponds to the boundary conditions of the problem, and is particularly called the *Dirichlet map* (*Neumann map*, resp.) for Dirichlet (Neumann, resp.) boundary control/noise. λ is a real number also associated with the boundary conditions. The aforementioned terms make the differential form of (4.27) only formal and require appropriate handling (e.g., when proving convergence of stochastic integrals, existence of optimal solutions, etc. - see [49]).

Studying optimal control problems constrained by dynamics as in (4.27) is rather challenging. Investigation of the HJB theory reveals that additional regularity conditions have to be imposed on certain terms. Especially when Dirichlet boundary noise is considered, proving even convergence of (4.27) becomes nontrivial. A few papers have attempted to tackle such problems and have dealt with specific formulations and SPDEs, such as the stochastic heat equation on the positive half-line. For further details, the reader can refer to the discussion in [49, Section 2.5 & Appendix C.5].

Nevertheless, in this section we will solely rely on a thermodynamics-based approach to find (sub)optimal policies for boundary control problems. Having obtained an abstract formulation of the dynamics constraints, we are able to use the variational optimization

methodology of the previous section. In particular, given a state cost functional J, we will attempt to approach its lower bound, as this is given in (4.30) under the corresponding path measures. Towards this goal, we will associate controlled and uncontrolled dynamics via the random variable:

$$\hat{V}(t) := V(t) - \int_0^t \psi(s)\mathrm{d}s, \tag{4.28}$$

with $\psi = B^{-1}GU$. $\hat{V}$ will define a cylindrical Wiener noise on the boundary under a measure $\mathbb{Q}$ that satisfies: $\mathrm{d}\mathbb{Q} = \exp\big(\int_0^T \langle\psi(s), \mathrm{d}V(s)\rangle_{\Lambda_0} - \frac{1}{2}\int_0^T ||\psi(s)||^2_{\Lambda_0}\mathrm{d}s\big)\mathrm{d}\mathbb{P}$. This change of measure was also utilized in reference [50] for studying solutions of SPDEs with noise and control occurring in the boundary conditions. By parameterizing the boundary controls as in (4.12), one can apply the same approach as in the previous section to numerically solve stochastic, boundary control problems. Notice that the corresponding inner products are taken here with respect to the Hilbert space Λ on the boundary.

4.1.4 Numerical Results

In this section we provide results on distributed and boundary control of semilinear SPDEs using the information theoretic variational approach. Following is a brief description on each experiment for 3 different SPDEs.

Distributed Stochastic Control

The 1-D stochastic Nagumo equation

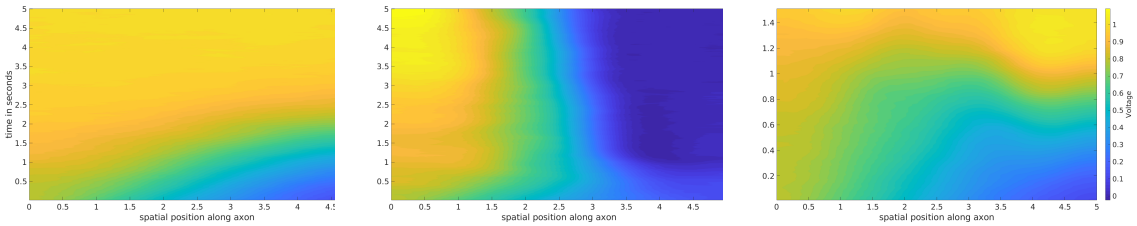
The stochastic Nagumo equation with homogeneous Neumann boundary conditions is a reduced model for wave propagation of the voltage in the axon of a neuron [51]. By simulating the deterministic Nagumo equation with a certain set of parameters, we observed that it requires 5 seconds to completely propagate the voltage across the axon (Fig.4.1a). We tested our proposed infinite-dimensional controller on two tasks: (1.) accelerating the

voltage propagation and (2.) suppressing the voltage propagation across the axon with time horizons of 1.5 seconds and 5 seconds respectively. Notice that the horizon for task (2.) is same as that required for the uncontrolled dynamics to propagate voltage across the axon. Towards the end of the horizon, the task gets harder as the controller pushes back against the dynamics. The controller was tested in both an open-loop and closed-loop (i.e. model predictive control or MPC) setting (Fig.4.1b and Fig.4.1c). For generating the open-loop control trajectory, 200 optimization iterations were performed wherein each iteration used 200 sampled trajectories, while for MPC, at each timestep, 10 optimization iterations were performed using 100 sampled trajectories. The performance of both controllers were compared by averaging the voltage profiles for 2^{nd}-half of each time horizon and repeated over 128 trials (see Figures 4.1d and 4.1e). A summary of the experimental results (in the region along the desired profile) in given in the table below. These plots also show the chosen locations of the actuator centers and their 1σ region of influence along the axon. Clearly, the closed loop controller outperforms the open-loop controller, which is evident in the zoomed-in subplots, where it minimizes the variance of the voltage profiles about the desired profile for both tasks.

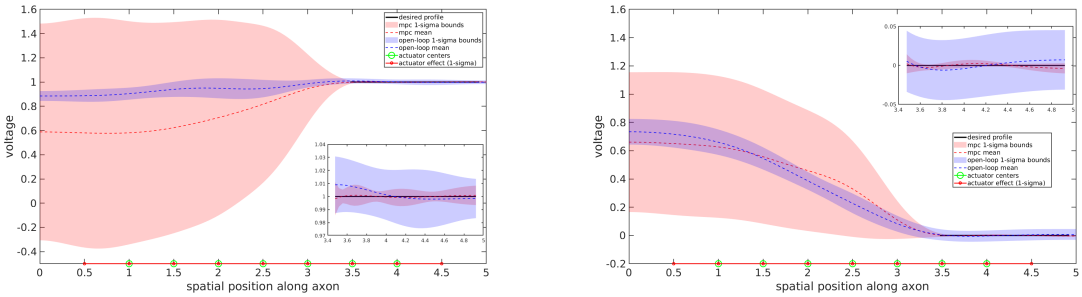
Task	**Acceleration**		**Suppression**	
	MPC	open-loop	MPC	open-loop
RMSE	$6.605e^{-4}$	0.0042	0.0021	0.0048
Avg. σ	0.0059	0.0197	0.0046	0.0389

The 1-D stochastic Burgers equation

Next, we consider the 1-D stochastic Burgers equation (viscous version) with non- homogeneous Dirichlet boundary conditions. This advection-diffusion equation with random forcing, has been studied as a simple model for turbulence ([52] and [53]). In our experiment, we tasked the proposed controller to achieve and maintain a desired velocity profile along the spatial domain at specific locations. In trying to do so, the controller has to overcome both the transport and diffusive nature of the uncontrolled dynamics. The



(a) Uncontrolled Dynamics (b) Suppressed Activity - MPC (c) Accelerated Activity - MPC



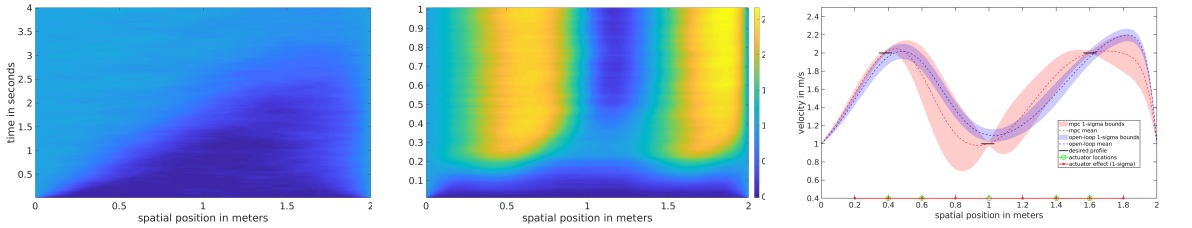
(d) Nagumo SPDE (Acceleration task) (e) Nagumo SPDE (Suppression task)

Figure 4.1: **Infinite dimensional control of the Nagumo SPDE**. Spatio-temporal profiles for: (a) uncontrolled spatio-temporal evolution for 5 seconds, (b) suppressed activity with MPC for 5 seconds and (c) accelerated activity with MPC within 1.5 seconds. The sub-plots (d) and (e) are voltage profiles averaged over the 2^{nd}-half of each time horizon over 128 trials. Moreover, we emphasize the performance around the desired profiles as the cost function is designed to focus on these spatial regions and ignore the rest.

uncontrolled spatio-temporal evolution, under non-homogeneous Dirichlet boundary conditions (non-zero constant flow velocity at the boundaries), is depicted in Fig. 4.2a, which shows the gradual build-up of velocity across the domain to an almost constant velocity profile (almost due to noise perturbations) starting from a zero-velocity profile. Similar to the Nagumo experiments, both open-loop and MPC controllers were implemented and tested (Fig. 4.2b) and their performances compared by averaging the voltage profiles for 2^{nd}-half of each time horizon and repeated over 128 trials (Fig. 4.2c). The time horizon considered for this task was 1 second. For open-loop, 100 optimization iterations and 100 sampled trajectories per iteration were used, while for MPC, at each timestep, 10 optimization iterations were performed using 100 sampled trajectories. Again, the effectiveness of the closed loop controller is obvious around the desired profile regions where the variance

and error of the averaged profiles is lower as compared to those of the open-loop controller.

	RMSE			Average σ		
Targets	**left**	**center**	**right**	**left**	**center**	**right**
MPC	0.0344	0.0156	0.0132	0.0309	0.0718	0.0386
Open-loop	0.0820	0.1006	0.0632	0.0846	0.0696	0.0797

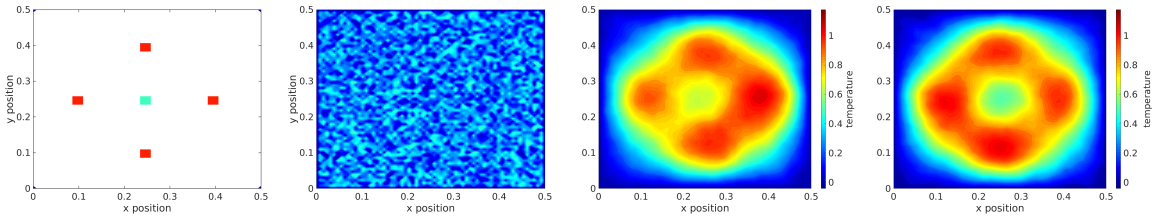


(a) Uncontrolled dynamics (b) MPC results (c) Average performance over trials

Figure 4.2: **Infinite dimensional control of the 1-D Burgers SPDE**. Above plots show: (a) spatio-temporal evolution of the uncontrolled 1-D Burgers SPDE with space-time white-noise, (b) spatio-temporal evolution of 1-D Burgers SPDE using MPC and (c.) velocity profiles averaged over the 2^{nd}-half of each time horizon over 128 trials.

The 2-D stochastic Heat equation

Next, we demonstrate control of the stochastic heat equation in 2D with homogeneous Dirichlet boundary conditions. We consider distributed control in form of actuators at specific locations on the 2D spatial domain. This scenario can be thought of as an insulated metallic plate (i.e. no interaction with the environment) with heaters at specific locations and the edges of the plate constantly freezing. The goal is to achieve and maintain a desired temperature profile as shown in Fig. 4.3a. Similar to previous experiments we implemented both open-loop and closed-loop controllers, but only emphasize on closed-loop control here. Starting from a random initial temperature profile as in Fig. 4.3b, and using a time horizon of 1 second, the MPC controller is able to achieve the desired temperature profile towards the end of the time horizon as shown in Fig. 4.3d. The MPC controller used 5 optimization iterations at every timestep and 25 sampled trajectories.



(a) Desired Profile (b) Random Initial Profile (c) Profile: Half-way (d) Profile at the End

Figure 4.3: **Infinite Dimensional control of the 2D-Heat SPDE** under homogeneous Dirichlet boundary conditions. Snapshots of temperature profiles: (a) desired profile, (b) initial random profile, (c) profile half way through the experiment and (d) profile at the end of experiment.

Boundary Stochastic Control

Boundary control problems are commonly found in physics and engineering disciplines, see e.g. [49]. To validate the approach of subsection 4.1.3 above, we will consider the stochastic 1-D heat equation under Neumann boundary conditions. Our goal is to track a time-varying profile by controlling the temperature only at the boundary points. The sub-optimal solution we obtain remains close to the task-specific desired profile (magenta surface), as illustrated in Fig. 4.4. The associated controls acting on the two boundary points are depicted in Fig. 4.5. These numerical results correspond to our MPC scheme with 10 optimization iterations and a standard quadratic cost function. Regarding the discretization method, we used finite differences with 130 nodes to propagate the SPDE forward in time [51].

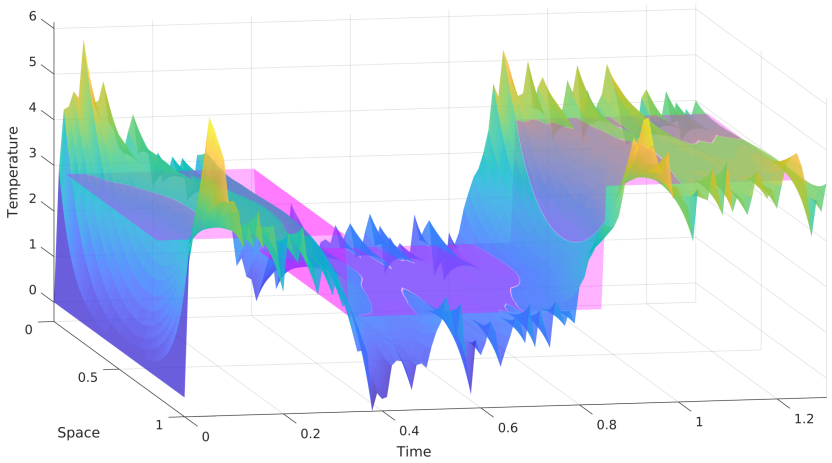


Figure 4.4: **Boundary control of stochastic 1-D heat equation.** The obtained temperature profile of a 1-D rod is illustrated over time. The magenta surface corresponds to the task-specific desired profile.

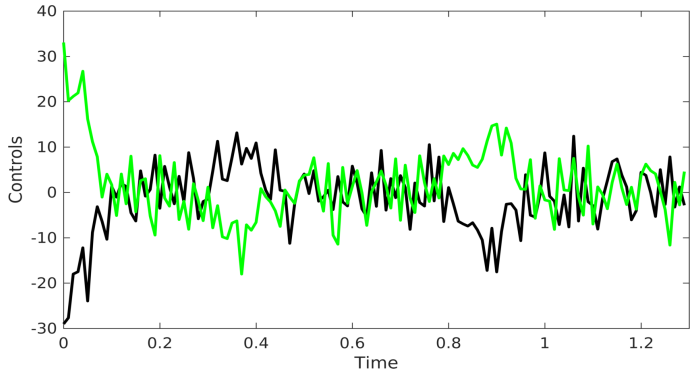


Figure 4.5: **Boundary control of stochastic 1-D heat equation.** Obtained control inputs entering through Neumann boudary conditions.

4.2 Variational Optimization Based Reinforcement Learning for Infinite Dimensional Stochastic Systems

In this section, we present a reinforcement learning framework in Hilbert spaces based on variational optimization and importance sampling for SPDEs. The resulting algorithm, called IDVRL, incorporates explicit feedback of the entire SPDE and allows for arbitrary non-linear policies such as Feed-forward Neural Networks (FNNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Moreover, we introduce a technique to handle numerical integration of policy networks over spatial domains which we call $SparseForwardPass$ for FNN and CNN policies, enabling scalability to 2D and 3D problems.

Consider the general semi-linear controlled SPDE given by

$$\mathrm{d}X = \big(\mathcal{A}X + F(t,X)\big)\mathrm{d}t + G(t,X)\big(\mathbf{\Phi}(t,X,\mathbf{x};\mathbf{\Theta}^{(k)})\mathrm{d}t + \frac{1}{\sqrt{\rho}}\mathrm{d}W(t)\big), \tag{4.29}$$

where $X(t) \in \mathcal{H}$ is the state of the system which evolves on the Hilbert space $\mathcal{H}$, the linear and nonlinear operators $\mathcal{A} : \mathcal{H} \to \mathcal{H}$ and $F(t,X) : \mathbb{R} \times \mathcal{H} \to \mathcal{H}$ (resp.) are uncontrolled drift terms, $\mathbf{\Phi}(t,X,\mathbf{x};\mathbf{\Theta}^{(k)}) : \mathbb{R}\times\mathcal{H}\times\mathbb{R}^3 \to \mathcal{H}$ is the nonlinear control policy parameterized by $\mathbf{\Theta}^{(k)}$ at the k^{th} iteration, $\mathrm{d}W(t) : \mathbb{R} \to \mathcal{H}$ is a Cylindrical spatio-temporal noise process (i.e. space-time white noise), and $G(t,X)$ is nonlinearity that affects both the Cylindrical noise and the control. It is used to incorporate the effects of actuation on either the field (distributed) or at the boundaries.

Define the uncontrolled and controlled probability measures associated with (4.2) as $\mathcal{L}$ and $\tilde{\mathcal{L}}$, respectively. These measures roughly describe the probabilistic evolution of the system, with the probability density function as a finite dimensional analog. In this case,

(2.8) takes the form [17]

$$-\frac{1}{\rho}\log\mathbb{E}_{\mathcal{L}}\Big[\exp(-\rho J)\Big]=\min_{\mathcal{U}(\cdot,\cdot)}\Big[\mathbb{E}_{\tilde{\mathcal{L}}}\left(J\right)+\frac{1}{\rho}D_{KL}(\tilde{\mathcal{L}}\,\big|\big|\,\mathcal{L})\Big], \tag{4.30}$$

where $J=J(X)$ can be viewed as an arbitrary state cost function. The associated "Work" and "Entropy" terms that minimize this expression describe a minimum "energy"[2] measure. Sampling from this measure would simultaneously minimize state cost and the KL-divergence between the controlled and uncontrolled distributions, which in this case is roughly interpreted as control effort. The measure that optimizes (4.30) is the so-called Gibbs measure

$$\mathrm{d}\mathcal{L}^{*}=\frac{\exp(-\rho\mathcal{J})\mathrm{d}\mathcal{L}}{\mathbb{E}_{\mathcal{L}}\big[\exp(-\rho\mathcal{J})\big]}. \tag{4.31}$$

While it is not known how to sample directly from (4.31), the goal of variational optimization methods is to incrementally reduce the distance (defined in the KullbackLeibler divergence sense) between the controlled distribution $\tilde{\mathcal{L}}$ and the optimal measure (4.31). We formulate our variational minimization problem as

$$\boldsymbol{\Theta}^{*}=\underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\,D_{KL}(\mathcal{L}^{*}||\tilde{\mathcal{L}})=\underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\left[\int\log\Big(\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\tilde{\mathcal{L}}}\Big)\frac{\mathrm{d}\mathcal{L}^{*}}{\mathrm{d}\mathcal{L}}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\tilde{\mathcal{L}}}\mathrm{d}\tilde{\mathcal{L}}\right]=\underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\;\mathcal{L}$$

Finally, we introduce a version of Girsanov's Change of Measure theorem between the uncontrolled and controlled processes, resulting in the so-called Radon-Nikodym derivative given as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\tilde{\mathcal{L}}}=$$
$$\exp\Big\{-\sqrt{\rho}\int_{0}^{T}\Big\langle\boldsymbol{\Phi}(t,X,\mathbf{x};\boldsymbol{\Theta}^{(k)}),\mathrm{d}W(t)\Big\rangle-\rho\frac{1}{2}\int_{0}^{T}\Big\langle\boldsymbol{\Phi}(t,X,\mathbf{x};\boldsymbol{\Theta}^{(k)}),\boldsymbol{\Phi}(t,X,\mathbf{x};\boldsymbol{\Theta}^{(k)})\Big\rangle\mathrm{d}t\Big\}. \tag{4.32}$$

[2]The term energy here is used loosely to describe the landscape for work and entropy

Plugging in (4.31) and (4.32) (for importance sampling), the loss-function L becomes

$$L = \mathbb{E}_{\tilde{\mathcal{L}}}\Bigg[\underbrace{\frac{\exp(-\rho\tilde{J})}{\mathbb{E}_{\tilde{\mathcal{L}}}\big[\exp(-\rho\tilde{J})\big]}}_{ImportanceWeight}\Bigg(-\sqrt{\rho}\int_0^T \underbrace{\Big\langle \mathbf{\Phi}(t,X,\mathbf{x};\mathbf{\Theta}^{(k)}), \mathrm{d}W(t)\Big\rangle}_{NoiseInnerProduct} \\ -\frac{1}{2}\rho\int_0^T \underbrace{\Big\langle \mathbf{\Phi}(t,X,\mathbf{x};\mathbf{\Theta}^{(k)}), \mathbf{\Phi}(t,X,\mathbf{x};\mathbf{\Theta}^{(k)})\Big\rangle}_{PolicyInnerProduct}\mathrm{d}t\Bigg)\Bigg], \tag{4.33}$$

where $\tilde{J}$ is defined by

$$\tilde{J} = \underbrace{J}_{StateCost} + \frac{1}{\sqrt{\rho}}\int_0^T \underbrace{\Big\langle \mathbf{\Phi}(t,X,\mathbf{x};\mathbf{\Theta}^{(k)}), \mathrm{d}W(t)\Big\rangle}_{NoiseInnerProduct} + \\ \frac{1}{2}\int_0^T \underbrace{\Big\langle \mathbf{\Phi}(t,X,\mathbf{x};\mathbf{\Theta}^{(k)}), \mathbf{\Phi}(t,X,\mathbf{x};\mathbf{\Theta}^{(k)})\Big\rangle}_{PolicyInnerProduct}\mathrm{d}t. \tag{4.34}$$

The loss-function L exponentiates the cost of the system trajectories, evaluated by $\tilde{J}$, to produce a weighted average of the mixed control-noise term and the quadratic control term. We minimize this loss via Stochastic Gradient Descent (SGD). The resulting Variational RL with learn rate γ is an incremental update of the form

$$\mathbf{\Theta}^{(k+1)} = \mathbf{\Theta}^{(k)} - \gamma\nabla_{\mathbf{\Theta}}L. \tag{4.35}$$

Our proposed approach uses an arbitrary non-linear feedback policy and produces an SGD-based minimization that can leverage well-known backprop-based algorithms such as ADA-Grad [54] and ADAM [55].

Although the state may be described by an infinite-dimensional vector in a Hilbert space, many physical realizations of actuation are defined on finite subspaces. The above derivation keeps $\mathbf{\Phi}$ as mapping into the Hilbert space, insinuating that the actuation may

be distributed everywhere and infinite-dimensional. However, the goal of this work is to ultimately use finite-action policy networks to control (4.2). As such, we refine $\mathbf{\Phi}$ as

$$\mathbf{\Phi}(t, X, \mathbf{x}; \mathbf{\Theta}^{(k)}) = \mathbf{m}(\mathbf{x})^\top \varphi(X; \mathbf{\Theta}^{(k)}), \tag{4.36}$$

where $\varphi(X; \mathbf{\Theta}^{(k)}) : \mathcal{H} \to \mathbb{R}^N$ is a finite policy network with N control outputs representing N distributed (or boundary) actuators. The function $\mathbf{m}(\mathbf{x}) : D \to \mathbb{R}^N \times \mathcal{H}$ represents the effect of the finite actuation on the infinite-dimensional field, where D is the domain of the finite spatial region. Some examples of $\mathbf{m}(\mathbf{x})$ are Gaussians-like exponential functions with mean centered at the actuator location (for distributed control) and indicator functions (for boundary control).

4.2.1 Algorithm and Network Architecture

The above derivation provides a mathematical framework for updating the weights of a policy network in a Reinforcement Learning (RL) setting. In order to implement it as an algorithm, data must be generated either from a physics-based or data-based model, or from interactions with a real system. Notice that since the only term from the dynamics to appear in (4.33), (4.34) is the Cylindrical noise term $\mathrm{d}W$, there is no need to have an explicit SPDE model. As a result, any black-box methods that incorporate spatio-temporal stochasticity can be used to generate sample trajectories of the system.

The above derivation introduces a unique problem for our proposed reinforcement learning framework that has not been addressed in prior work. Each inner product in Hilbert space in (4.33), (4.34) represents a spatial integration over a finite region D. To the knowledge of the authors, integration over a policy network has not been attempted to date. However in this work, we integrate spatially over the input to the network. Consider the inner product indicated as *PolicyInnerProduct*. The representation of this inner product

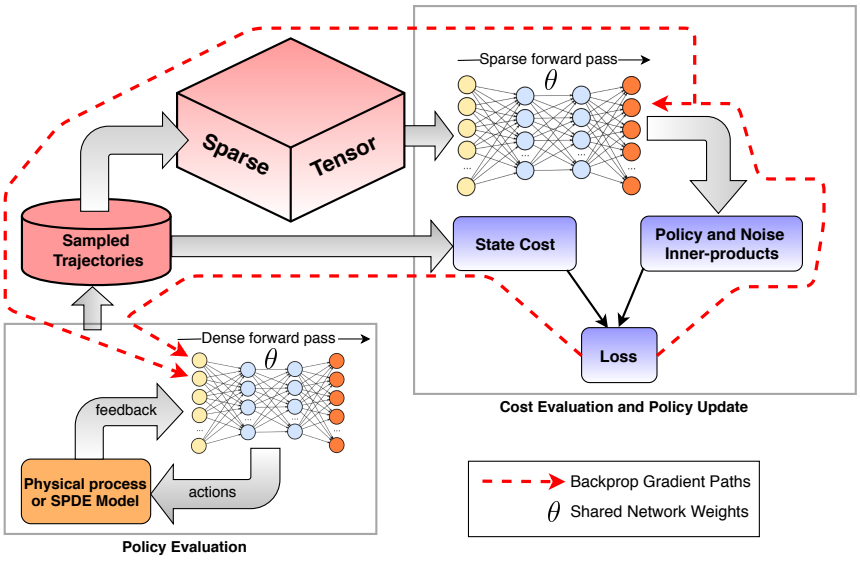


Figure 4.6: Block diagram of computational graph for the IDVRL algorithm.

as a spatial integration takes the form

$$\int_0^T \left\langle \mathbf{\Phi}(X, \mathbf{x}; \mathbf{\Theta}^{(k)}), \mathbf{\Phi}(X, \mathbf{x}; \mathbf{\Theta}^{(k)}) \right\rangle \mathrm{d}t = \tag{4.37}$$

$$\int_0^T \iint_D \varphi(X(t,x,y); \mathbf{\Theta}^{(k)})^\top M(x,y) \varphi(X(t,x,y); \mathbf{\Theta}^{(k)}) \mathrm{d}x\mathrm{d}y\mathrm{d}t$$

$$= \int_0^T \sum_{j=1}^{\infty} \varphi(X(e_j); \mathbf{\Theta}^{(k)})^\top M(e_j) \varphi(X(e_j); \mathbf{\Theta}^{(k)}) \mathrm{d}t, \tag{4.38}$$

where $D \subseteq \mathbb{R}^2$ is the problem domain, $\{e_j \in \mathcal{H} : j = 0, 1, 2, \dots\}$ forms an orthonormal basis over $\mathcal{H}$, and $M(\mathbf{x}) = \mathbf{m}(\mathbf{x})\mathbf{m}(\mathbf{x})^\top$. After discretization on a 2D grid, the basis becomes a finite set $\{e_j \in \mathbb{R}^{J^2} : j = 0, 1, 2, \dots\}$, where each element is a one-hot vector. Thus, evaluating the spatial integral is reduced to summing up forward passes through the policy network with each pixel considered individually. Note that this spatial integration approach is agnostic to choice of discretization scheme.

Spatially integrating over the policy network is a memory intensive task, where the storage becomes (J^2, J, J) for each sample over the time horizon. However, given that the basis elements of each (J, J) "image" have only one activated "pixel", the resulting tensor is tremendously sparse. As such, each layer's activation can be computed with a sparse matrix multiplication, resulting in what we call a $SparseForwardPass$ method that is

not memory intensive for relatively large 2D problems. This can be applied to both FNNs and CNNs. For CNNs, activation can be achieved by matrix multiplication with a Toeplitz matrix constructed from the filter coefficients [56].

A summary of our architecture is depicted in (4.6). A policy network with initialized weights is passed through a model or physical realization of the system to produce state trajectories, which are used to compute a state cost as well as a sparse tensor that is used to compute the inner products in (4.33), (4.34) in a memory and time-efficient manner. Finally the loss is computed and passed to a gradient-based optimization algorithm. This approach is independent of specific policy network architecture used, which can often be problem dependent. In this work we used two different networks: a FNN for 1D SPDE and a CNN for 2-dimensional (2D) SPDE.

The resulting IDVRL algorithm is shown in (4), wherein subscript implies an element of the corresponding vector. The input terms are time horizon (T), number of iterations (K), number of rollouts (R), initial state (X_0), number of actuators (N), noise variance (ρ), time discretization (Δt), actuator locations (μ), actuator variance (σ_μ, for distributed control cases), and initial network parameters ($\mathbf{\Theta}^{(0)}$). We note that the function $GradientOptimize(L, \mathbf{\Theta}^{(k)})$ represents the update from (4.35). As mentioned above, this is handled by any variant of SGD, which performs backpropagation through the network. The computational graph of the proposed algorithm has multiple backprop paths, as shown by the dotted red line in (4.6). For more information on $SampleNoise()$, refer to [51, Chapter 10].

4.2.2 Simulation Results and Discussion

We applied the IDVRL to reaching tasks for several SPDEs in simulation in both distributed and boundary control settings. In each reaching task, the policy has to control the system to achieve a desired profile in certain parts of the spatial domain. These simulated experiments were developed via computational graphs implemented in Tensorflow [57] to

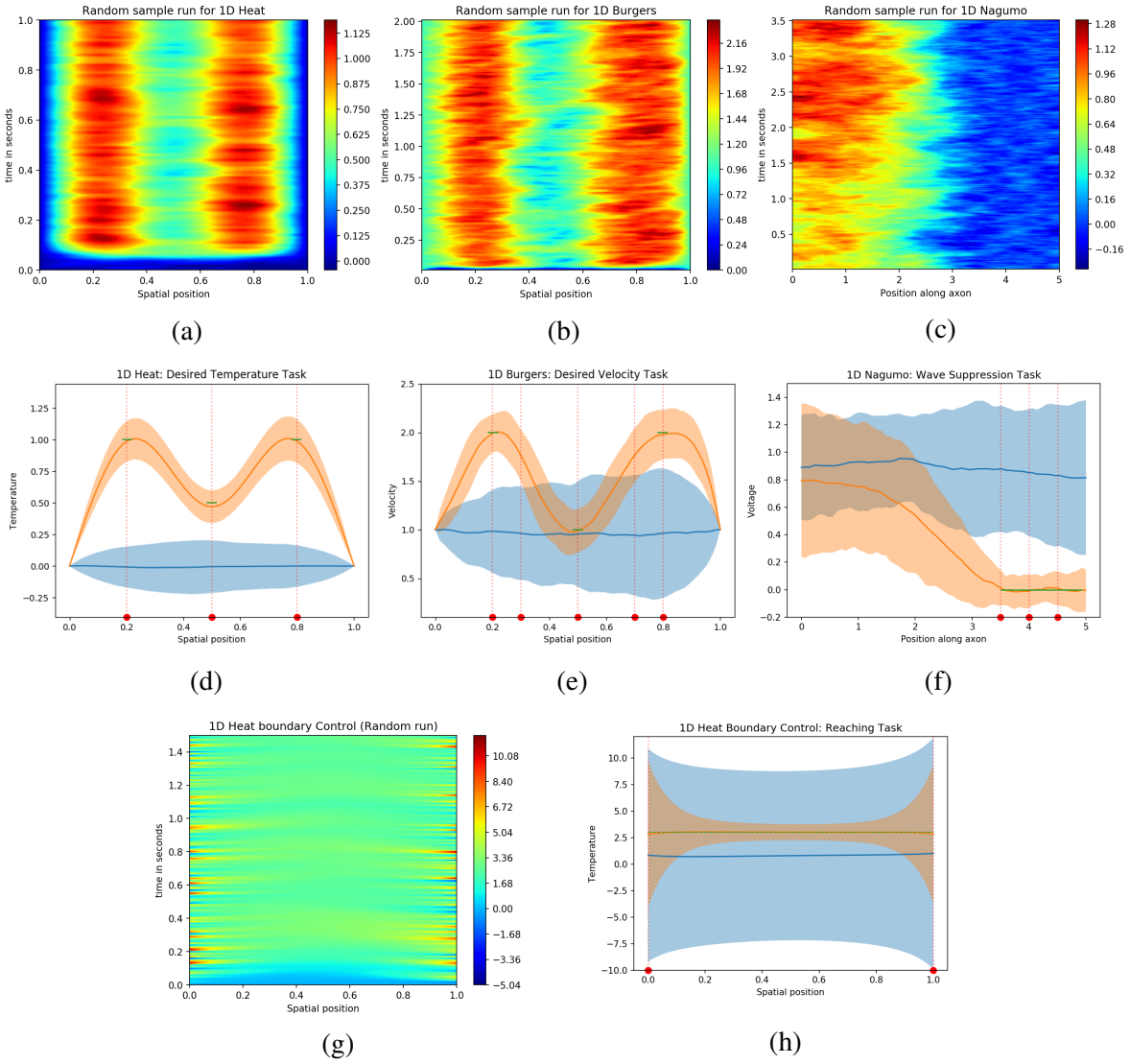


Figure 4.7: **Control of 1D SPDEs.** (a), (d), (g), (h) correspond to the Heat SPDE, (b), (e) to Burgers SPDE, and (c), (f) to Nagumo SPDE. In (d), (e), (f), (h) blue represents *mean uncontrolled profiles*, orange represents *mean controlled profiles* using the trained policy network, green represents *desired values* in certain spatial regions, and red represents *locations of actuator centers*. The mean and variance statistics are gathered over 200 rollouts. (a), (b), (c), (g) depict a randomly selected trial run to emphasize the presence of spatio-temporal stochasticity. (a-f) depict results for distributed control of SPDEs and (g-h) depict results for boundary control of a SPDE.

Algorithm 4 Infinite Dimensional Variational Reinforcement Learning

Function: $\boldsymbol{\Theta}^* = \textbf{\textit{OptimizePolicyNetwork}}(T, K, R, X_0, N, \rho, \Delta t, \mu, \sigma_\mu, \boldsymbol{\Theta}^{(0)})$
Compute $\mathbf{m}(\mathbf{x}), M(\mathbf{x}) \; \forall \; \mathbf{x} \in D$
for $k = 1$ to K **do**
  for $r = 1$ to R **do**
    for $t = 1$ to T **do**
      $\mathrm{d}W_t \leftarrow SampleNoise()$
      $X_t \leftarrow Propagate(X_{t-1}, \boldsymbol{\Theta}^{(k)}, \mathrm{d}W_t)$ via (4.2)
      $J_r \leftarrow J_r + StateCost(X_t)$
      $S_t \leftarrow SparseForwardPass(\boldsymbol{\Theta}^{(k)}, X_t)$
      $N_t \leftarrow NoiseInnerProduct\big(S_t, \mathrm{d}W_t, \mathbf{m}(\mathbf{x})\big)$
      $P_t \leftarrow PolicyInnerProduct\big(S_t, M(\mathbf{x})\big)$
    end for
    $P, \; N \leftarrow Sum(P_t), \; Sum(N_t)$
    $\tilde{J}_r \leftarrow \tilde{J}(P, N, J_r)$
  end for
  $W \leftarrow ImportanceWeight(\tilde{J})$
  $L \leftarrow ComputeLoss(P, N, W)$ via (4.34)
  $\boldsymbol{\Theta}^{(k+1)} \leftarrow GradientOptimize(L, \boldsymbol{\Theta}^{(k)})$
end for

leverage GPU parallelization for training as well as sparse linear algebra operations for $SparseForwardPass$. The data for training the policies was generated by simulating the SPDEs using centered finite-difference approximation for the spatial derivatives on a 1D or 2D grid and a semi-implicit Euler scheme for discretization of the time derivatives. For detailed explanation on these schemes, we refer the reader to [51, Chapters 3 and 10]. For 1D simulations, we used an Alienware laptop with an Intel Core i9-8950HK CPU @ 2.9GHz$\times$12, 32 GB RAM and a NVIDIA GeForce GTX 1080 graphics card. On average, Tensorflow-GPU required around 16 minutes of training time for 1000 iterations. For the 2D simulation, we used Tensorflow-CPU, due to insufficiency of VRAM, which required around 12 hours of training time for 1000 iterations. The code and videos for these experiments are available online [3].

Figure 4.7 (a) and (d) depict the results of the IDVRL algorithm on a task of controlling the 1D heat SPDE with homogeneous Dirichlet boundary conditions. The goal of the task

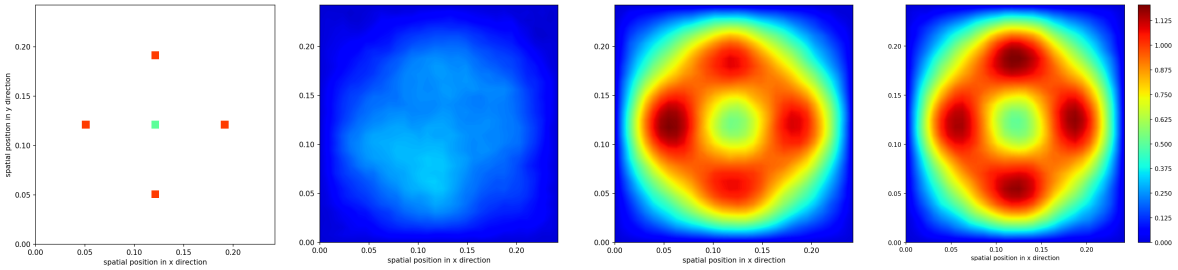
[3] Code: https://github.gatech.edu/eevans41/spde_explicit_feedback_RL, Video: https://youtu.be/6tmky59xhp4

is to raise and maintain the temperature to $T = 1$ at regions around $x = 0.2$ and $x = 0.8$, and $T = 0.5$ at a region around $x = 0.5$. Figure 4.7a) shows the temperature contours of a single realization of the completed task and (4.7)d) shows the mean controlled and uncontrolled trajectories at the final time with a 2-σ variance shaded in the corresponding color. The boundary conditions fixed the endpoints to a temperature of $T = 0$, as shown.

Figure 4.7, (b) and (e) depict the results of the IDVRL algorithm on the task of controlling the 1D Burgers SPDE with non-homogeneous Dirichlet boundary conditions. In this task the goal is to reach a desired velocity in the medium at given locations. This is challenging given the nonlinear advection behavior of the system in addition to the pure diffusion behavior shown in the 1D heat SPDE task. The advection-diffusion creates an apparent rightwards wave-front that must be accounted for by the policy network in order to achieve the task. Given the increased difficulty of the problem, we added actuators, as indicated by vertical red dotted lines. Despite the added actuators, the task remains severely under-actuated.

Figure 4.7, (c) and (f) depict the IDVRL algorithm on the task of controlling the 1D Nagumo SPDE with homogeneous Neumann boundary conditions. As noted earlier, the Nagumo SPDE represents voltage travelling across the axon of a neuron in the brain. The goal of this task is to suppress the voltage from travelling across the axon. Voltage near 1.0 indicates the voltage has travelled across, and in this suppression task, we seek to keep the voltage at the right end of the axon at $V = 0$. The Nagumo SPDE has a 3rd order nonlinearity. For this task, we supplied the system with only three actuators near the right end, where voltage must be suppressed.

For the next task, we scaled the IDVRL algorithm to two-dimensional problems. With this task we attempt to control the 2D Heat SPDE with homogeneous Dirichlet boundary conditions with a CNN policy network. The goal of this task it to raise the temperature in five regions. The desired temperature at the four outer regions is $T = 1$ and the desired temperature at the center region is $T = 0.5$. Figure 4.8 depicts a single realization of the



(a) (b) (c) (d)

Figure 4.8: **Control of the 2D Heat SPDE**. (a) shows the desired profile patches and actuator locations for the reaching task. The next three plots show time snapshots from a randomly selected instance of an optimized policy applied to the system. (b) shows the start profile(b), (c) shows half-way through, and (d) shows the end profile. The color-bar depicts the range of temperatures in the simulated field.

controlled task under a significant amount of noise with five actuators.

In contrast to the previous tasks where actuators are distributed in the field, Figure 4.7h) depicts a *boundary* control task, where the actuator controls the boundary condition. The Radon-Nikodym derivative exists for the case of boundary control of semi-linear SPDEs with boundary noise [52], and we demonstrate that our method similarly extends to this case. The task here is similar to the first case, where the policy network is tasked with reaching a desired value of $T = 3$.

We invite the interested reader to refer to our supplementary material for specific details on each of our simulations such as cost functions, hyper-parameter values, neural network parameters and videos comparing controlled and uncontrolled SPDEs.

Throughout our simulated experiments, especially for distributed control tasks, we found that the algorithm is not sensitive to the majority of our parameters. We noted that a useful heuristic in applying the algorithm to new problems without having to tune the parameters was to ensure that the starting loss function was not very close to zero (i.e. 1e-10). Despite a large variance of noise that we typically applied to our systems ($\rho = 10$), the optimization algorithm was able to converge in under 1000 iterations for 1D problems and under 2000 iterations for 2D problems.

On the whole, even though injecting higher variance noise into the system inherently

makes the control task much more challenging, high variance noise is useful in our algorithm for exploration over rollouts at each iteration. As such, there is an inverse relationship for a given convergence behavior between variance in the noise and number of rollouts.

There are also several interesting behaviors that the IDVRL algorithm demonstrates. First, we noticed that often times throughout optimization, the loss would decrease as desired, but state cost would temporarily increase, before decreasing more dramatically after some number of iterations. This indicates that there may not be a strictly proportional relationship between loss function and state cost. Indeed a lower state cost implies that the task is being accomplished, yet a trend of decreasing loss function indicated that when there was a temporary increase in state cost, the IDVRL algorithm may have been pushing the network parameters out of a local minimum towards better task performance in later iterations.

Table 4.1: Variable notation

x^k, $\boldsymbol{x}$	States
u^k, $\boldsymbol{u}$	Control inputs
θ^k, $\boldsymbol{\theta}$	Control parameters
$\chi^k(\cdot)$, $\chi(\cdot)$	Probability density function of $\boldsymbol{\theta}$
ρ^k, $\boldsymbol{\rho}$	Parameters of p.d.f. $\chi(\boldsymbol{\theta})$

4.3 Trajectory optimization using stochastic approximation

4.3.1 Unconstrained stochastic optimization

Derivation for discrete dynamic systems

Our sampling control method has been inspired by the work in [11]. Therein, the authors attempt to minimize costs without structural properties such as differentiability and convexity. Here, we modify the particular framework to account for discrete-time Markovian dynamics.

Let us begin with the stochastic, discrete-time optimal control problem:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \mathbb{E}\big[J\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big)\big] \\ &\text{s.t. } x^{k+1} = f^k(x^k, u^k(x^k;\theta^k)), \;\; k = 0, 1, ..., H. \end{aligned} \tag{4.39}$$

Here, $x^k \in \mathbb{R}^n$, $u^k \in \mathbb{R}^m$, $\theta^k \in \mathbb{R}^d$ denote the state, control input and associated control parameters, respectively, at the kth time instance. That is, the control inputs u^k will be parameterized by a set of parameter vectors θ^k, over which we will be optimizing the expected cost $\mathbb{E}[J]$. The stochastic transition dynamics f^k impose a probability density function (p.d.f.) for the next states, $x^{k+1} \sim p(x^{k+1}|x^k, u^k)$. We will denote the corresponding state, control and parameter sequences as $\boldsymbol{x} := ((x^0)^\top, ..., (x^H)^\top)$, $\boldsymbol{u} := ((u^0)^\top, ..., (u^{H-1})^\top)$ and $\boldsymbol{\theta} := ((\theta^0)^\top, ..., (\theta^{H-1})^\top)$, respectively.

Inspired by [11], we will introduce a sampling distribution for the control parameters $\boldsymbol{\theta}$, over which we will optimize the expected value of $J(\cdot)$. Let us denote by $\chi(\boldsymbol{\theta};\boldsymbol{\rho})$ this

distribution of $\boldsymbol{\theta}$ which will be parameterized by the vector $\boldsymbol{\rho} := ((\rho^0)^\top, ..., (\rho^{H-1})^\top)$. Then, we will aim to minimize:

$$\begin{aligned} &\min_{\boldsymbol{\rho}} \mathbb{E}\big[J\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}; \boldsymbol{\theta})\big)\big], \\ &\text{s.t.} \quad x^{k+1} = f(x^k, u^k(\theta^k)), \\ &\qquad \theta^k \sim \chi^k(\theta^k; \rho^k), \quad k = 0, 1, ..., H, \end{aligned} \tag{4.40}$$

We will henceforth assume that $\chi(\cdot)$ belongs to the exponential family of distributions [58].

To justify this approach, notice that the optimum cost of (4.40) is always an upper bound to the minimum cost in (4.39), associated with $\boldsymbol{\theta}_*$. Moreover, the two costs become equal when the entire probability mass of $\chi(\cdot)$ is concentrated on $\boldsymbol{\theta}_*$. Hence, it is intuitive to minimize the expected value of the original cost with respect to a distribution over $\boldsymbol{\theta}$.

Further, notice that we can allow J to be non-convex and even discontinuous, since we optimize with respect to $\boldsymbol{\rho}$ which only appears in $\chi(\cdot)$. Our controller, $u^k(\theta^k)$, also provides great flexibility since it can define a feedforward, feedback, or even non-linear policy. These advantages stem from our choice to optimize (4.40) not directly with respect to the control variables $\boldsymbol{\theta}$, but in terms of the parameters $\boldsymbol{\rho}$ of their sampling distributions. A sketch of the corresponding optimization methodology is described by Algorithm 5.

Algorithm 5 Methodology for stochastic optimization

Require: Initial parameters $\boldsymbol{\rho}$.

1: **while** $\boldsymbol{\rho}$ has not converged **do**
2: Sample candidate control parameters $\boldsymbol{\theta}$ from $\chi(\boldsymbol{\theta}; \boldsymbol{\rho})$ and evaluate their costs J
3: Use a gradient-based method to update $\boldsymbol{\rho}$ and thus $\chi(\boldsymbol{\theta}; \boldsymbol{\rho})$
4: **end while**

Example: One simple example of this formulation is the following: Suppose we parameterize our control with a linear policy $u^k = K^k x + c^k$, where now θ^k corresponds to $\{K^k, c^k\}$. If we consider each K^k and c^k to follow Gaussian distributions, then (4.40) is minimized over their corresponding means and variances.

To proceed, we will introduce a continuous *shape* function, $S : \mathbb{R} \to \mathbb{R}^+$, such that $S(y)$ is monotonically decreasing in y, $\forall y \in \mathbb{R}$. One example is $S(y) = \exp(-\kappa y)$ for some $\kappa > 0$, which we will also use in our simulations. We will also take a logarithmic transformation of the expected cost in (4.40) before performing minimization. We introduce these auxiliary transforms because (i) it has been shown that certain selections empirically improve numerical implementation [11], and (ii) is is easier to show connections with Path Integral control-related schemes. Therefore, we will attempt to solve the equivalent problem:

$$\begin{aligned}\max_{\boldsymbol{\rho}} \mathcal{L}(\boldsymbol{\rho}) &= \max_{\boldsymbol{\rho}} \ln\left(\mathbb{E}\left[S\big(J\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{\theta})\big)\big)\right]\right) \\ &= \max_{\boldsymbol{\rho}} \ln\left(\int S\big(J(\boldsymbol{\theta})\big)\Gamma(\boldsymbol{\theta}) \prod_{k=0}^{H} \chi^k(\theta^k; \rho^k) \mathrm{d}\theta^0 \cdots \mathrm{d}\theta^H\right).\end{aligned} \tag{4.41}$$

where we have dropped the implicit dependence of x^k and u^k on θ^k. The last line of (4.41) is obtained by assuming that the transition dynamics $f^k(\cdot)$ are Markovian, as well as imposing independence of θ^k's with respect to other terms. Hence, the distribution of a state/control sequence can be written as:

$$\begin{aligned}p(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}) &= p(x^0)p(u^0|x^0; \theta^0)\chi^0(\theta^0; \rho^0)p(x^1|u^0, x^0) \cdots \\ &p(u^{H-1}|x^{H-1}; \theta^{H-1})\chi^{H-1}(\theta^{H-1}; \rho^{H-1})p(x^H|u^{H-1}, x^{H-1})) \\ &= \Gamma(\boldsymbol{\theta}) \prod_{k=0}^{H} \chi^k(\theta^k; \rho^k),\end{aligned}$$

where $\Gamma(\cdot)$ is defined accordingly. Now, to maximize equation (4.41), we will have to compute the derivatives of $\mathcal{L}$ with respect to $\boldsymbol{\rho}$. Since each χ^k belongs to the exponential family, we can write [58]

$$\chi^k(\theta^k; \rho^k) = \exp((\rho^k)^\top T(\theta^k) - \phi(\rho^k)), \tag{4.42}$$

were $T(\theta^k)$ denotes the vector of sufficient statistics and $\phi(\rho^k) := \ln(\int \exp((\rho^k)^\top T(\theta^k))\mathrm{d}\theta^k)$. By pushing the gradient inside the integral in (4.41), one can explicitly compute [11]

$$\nabla_{\rho^k}\mathcal{L} = \mathbb{E}_{\mathcal{P}}[T(\theta^k)] - \mathbb{E}_{\chi^k}[T(\theta^k)], \tag{4.43}$$

where $\mathbb{E}_{\mathcal{P}}$ denotes expectation with respect to the "path" distribution defined by

$$\mathcal{P} := \frac{S\big(J(\{\theta^k\})\Gamma(\{\theta^k\})\prod_{k=0}^{H}\chi^k(\theta^k;\rho^k)}{\int S\big(J(\{\theta^k\})\Gamma(\{\theta^k\})\prod_{k=0}^{H}\chi^k(\theta^k;\rho^k)\mathrm{d}\theta^0\cdots\mathrm{d}\theta^H} \tag{4.44}$$

and $\mathbb{E}_{\chi^k}$ denotes expectation with respect to the distribution of parameters θ^k. Similarly, one can define higher-order derivatives. Based on the expressions above, a gradient ascent scheme for (4.41) simply reads:

$$(\rho^k)_{i+1} = (\rho^k)_i + \gamma_i\big(\mathbb{E}_{\mathcal{P}_i}[T(\theta^k)] - \mathbb{E}_{\chi_i^k}[T(\theta^k)]\big), \tag{4.45}$$

where i corresponds to the ith iteration of the algorithm and γ_i is a (possibly) iteration-dependent learning rate. The leftmost expectation above is computed through sampling, while the remaining term can be computed analytically. More complex updates can be designed in an analogous manner. By changing the natural parameters ρ in an iterative fashion, we expect to converge to a distribution of the control variables θ that minimize our cost.

Comparison with previous works

The developed optimization algorithm can be viewed as a generalization of stochastic control schemes derived from Path Integral control theory [16]. Therein, the dynamics are treated as a stochastic differential equation where Wiener noise usually enters through the control channel. To obtain a numerical algorithm, such works approximate the dynamics with an Euler-Maruyama scheme and proceed by minimizing the Kullback-Leibler of a

cost between uncontrolled and controlled dynamics. Specifically, the transition dynamics are restricted to the form: $f(x^k, \bar{u}^k) = F(x^k) + G(x^k)(\bar{u}^k + \xi^k)$, where F, G are properly defined matrices, $\xi^k \sim \mathcal{N}(0, \Delta t)$ and Δt is the time step. The update scheme from [17] reads:

$$(\bar{u}^k)_{i+1} = (\bar{u}^k)_i + \mathbb{E}_{\mathcal{Q}_i}[\xi^k], \tag{4.46}$$

with $\mathcal{Q} := \frac{\exp(-\kappa J(\{\theta^k\}) \prod_{k=0}^{H-1} p(x^{k+1}|u^k, \theta^k)}{\int \exp(-\kappa J(\{\theta^k\})) \prod_{k=0}^{H-1} p(x^{k+1}|u^k, \theta^k) \mathrm{d}x^0 \cdots \mathrm{d}x^H}$, $\kappa > 0$.

The above expressions can essentially be viewed as a specific case of (4.45). Indeed, this is true when $S(y) \equiv \exp(-\kappa y)$, $\theta^k \equiv u^k$, $\gamma = \sqrt{\sigma}$ and each θ^k is Gaussian with fixed variance. To see the latter, notice that the last term in (4.43) is a constant number with respect to the p.d.f. $\mathcal{Q}$. Hence, due to the Gaussianity of θ^k one has

$$\mathbb{E}_{\mathcal{Q}_i}[T(\theta^k)] - \mathbb{E}_{\chi_i^k}[T(\theta^k)] = \mathbb{E}_{\mathcal{Q}_i}[\sqrt{\sigma}\xi].$$

Similar expressions are given in works [16], where the policy has a specific form.

We stress again though that expressions (4.43), (4.45) hold for the statistics of *any type of parameterized policy*. In contrast, optimizing over generic policies by following the approach in [17] is not straightforward. Specifically, one will have to differentiate the change of measure between uncontrolled and controlled dynamics. Regularly, such computations cannot be carried out analytically, and are only specific to each candidate policy.

Similar update equations are also used within the Cross Entropy method [59], where only a set of elite sampled policies are used, whose number has to be pre-specified. We omit further details due to space limitations, and will make numerical comparisons with these methods in section 4.3.4.

4.3.2 Constrained sampling-based trajectory optimization

Sampling control with box constraints

We will now show an extension of the methodology above that accounts for box constraints on the control parameters

$$\boldsymbol{l} \leq \boldsymbol{\theta} \leq \boldsymbol{u}$$

in a direct fashion; that is, without casting them as soft constraints.

We recall that our approach to solving (4.39) was to iteratively sample candidate solutions from a distribution $\chi(\boldsymbol{\theta}; \boldsymbol{\rho})$, and then update the underlying distribution. The idea here is to constrain ourselves to truncated distributions over the specified region. In this way, all candidate solutions will be directly sampled within the acceptable domain.

In this direction, we will let the sampling distribution of the control parameters $\boldsymbol{\theta}$ have the form:

$$\chi(\boldsymbol{\theta}; \boldsymbol{\rho}) = \frac{\tilde{\chi}(\boldsymbol{\theta}; \boldsymbol{\rho})}{\int_{\boldsymbol{l}}^{\boldsymbol{u}} \tilde{\chi}(\boldsymbol{\theta}; \boldsymbol{\rho}) \mathrm{d}\boldsymbol{\theta}}, \tag{4.47}$$

where $\tilde{\chi}(\boldsymbol{\theta}; \boldsymbol{\rho})$ is of exponential type and $(\boldsymbol{l}, \boldsymbol{u}]$ denotes the hard control bounds[4]. Equation (4.47) implies that χ is the truncated distribution corresponding to $\tilde{\chi}$. We will consider

$$\tilde{\chi}(\boldsymbol{\theta}; \boldsymbol{\rho}) = \begin{cases} \exp(\boldsymbol{\rho}^\top T(\theta) - \phi(\boldsymbol{\rho})) & , \boldsymbol{\theta} \in (\boldsymbol{l}, \boldsymbol{u}] \\ 0 & , \text{otherwise} \end{cases}$$

where $\phi(\cdot)$ is a properly defined function [58]. It is easy to verify that (4.47) defines a proper distribution. Now denote for brevity the expected cost from (4.41) as $L := \mathbb{E}[S(J)]$. We will compute the gradient of $\ln L(\boldsymbol{\rho})$ when the expectation is computed with respect to the truncated distribution (4.47). Specifically, we will have

$$\nabla_{\boldsymbol{\rho}} \ln L(\boldsymbol{\rho}) = \frac{\int_{\boldsymbol{l}}^{\boldsymbol{u}} S(J(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}; \boldsymbol{\theta}))) \Gamma(\boldsymbol{\theta}) \nabla_{\boldsymbol{\rho}} \ln(\chi(\boldsymbol{\theta}; \boldsymbol{\rho})) \chi(\boldsymbol{\theta}; \boldsymbol{\rho}) \mathrm{d}\boldsymbol{\theta}}{\int_{\boldsymbol{l}}^{\boldsymbol{u}} S(J(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}; \boldsymbol{\theta}))) \Gamma(\boldsymbol{\theta}) \chi(\boldsymbol{\theta}; \boldsymbol{\rho}) \mathrm{d}\boldsymbol{\theta}}.$$

[4] In case the lower bound can be attained, we subtract a small positive number from $\boldsymbol{l}$.

From (4.47) one obtains

$$\nabla_{\boldsymbol{\rho}} \ln(\chi(\boldsymbol{\theta};\boldsymbol{\rho})) = \underbrace{\nabla_{\boldsymbol{\rho}}\big(\ln(\tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho}))\big)}_{\Gamma_1} - \underbrace{\nabla_{\boldsymbol{\rho}}\big(\ln\big(\int_{\boldsymbol{l}}^{\boldsymbol{u}} \tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho})\mathrm{d}\boldsymbol{\theta}\big)\big)}_{\Gamma_2},$$

where from the previous derivation we have

$$\Gamma_1 = T(\boldsymbol{\theta}) - \mathbb{E}_{\tilde{\chi}}[T(\boldsymbol{\theta})],$$

while

$$\Gamma_2 = \frac{\int_{\boldsymbol{l}}^{\boldsymbol{u}} \nabla_{\boldsymbol{\rho}} \tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho})\mathrm{d}\boldsymbol{\theta}}{\int_{\boldsymbol{l}}^{\boldsymbol{u}} \tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho})\mathrm{d}\boldsymbol{\theta}} = \frac{\int_{\boldsymbol{l}}^{\boldsymbol{u}} \nabla_{\boldsymbol{\rho}} \ln(\tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho}))\tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho})\mathrm{d}\boldsymbol{\theta}}{\int_{\boldsymbol{l}}^{\boldsymbol{u}} \tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho})\mathrm{d}\boldsymbol{\theta}} =$$
$$\frac{\int_{\boldsymbol{l}}^{\boldsymbol{u}} (T(\theta) - \mathbb{E}_{\tilde{\chi}}[T(\theta)])\tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho})\mathrm{d}\boldsymbol{\theta}}{\int_{\boldsymbol{l}}^{\boldsymbol{u}} \tilde{\chi}(\boldsymbol{\theta};\boldsymbol{\rho})\mathrm{d}\boldsymbol{\theta}} = \mathbb{E}_{\chi}[T(\theta)] - \mathbb{E}_{\tilde{\chi}}[T(\theta)].$$

Hence

$$\nabla_{\boldsymbol{\rho}} \ln(\chi(\boldsymbol{\theta};\boldsymbol{\rho})) = T(\boldsymbol{\theta}) - \mathbb{E}_{\chi}[T(\boldsymbol{\theta})],$$

and

$$\nabla_{\boldsymbol{\rho}} \ln L(\boldsymbol{\rho}) = \mathbb{E}_{\mathcal{P}}[T(\boldsymbol{\theta})] - \mathbb{E}_{\chi}[T(\boldsymbol{\theta})]. \tag{4.48}$$

where $\mathbb{E}_{\mathcal{P}}[T(\boldsymbol{\theta})]$ is computed numerically based on (4.44) and $\mathbb{E}_{\chi}[T(\boldsymbol{\theta})]$ can be computed analytically for certain p.d.f.'s, including the truncated normal distribution.

Sampling control with nonlinear state constraints

Suppose now we have the following generic, constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{\theta}} \ & \mathbb{E}\big[J\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big)\big] \\ \text{s.t. } & \mathbb{E}\big[g_i^k(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big] \leq 0, \ \ i = 1, ..., L \\ & \mathbb{E}\big[h_j^k(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big] = 0, \ \ j = 1, ..., D \\ & l^k \leq \theta^k \leq u^k, \ \ k = 0, ..., H, \end{aligned} \tag{4.49}$$

where we have omitted the dynamics constraints for simplicity. A standard approach for solving (4.49) with sampling control is to sample trajectories as in section 4.3.1 and simply assign high costs to infeasible rollouts [60]. In contrast, we will explore here *non-smooth penalty functions*.

First, let us use the methodology of sections 4.3.1 and 4.3.2 and transform (4.49) into

$$\begin{aligned} \min_{\boldsymbol{\rho}} \ & \mathbb{E}\big[J\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big)\big] \\ \text{s.t. } & \mathbb{E}\big[g_i^k(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big] \leq 0, \ \ i = 1, ..., L \\ & \mathbb{E}\big[h_j^k(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big] = 0, \ \ j = 1, ..., D \\ & \boldsymbol{\theta} \sim \chi(\boldsymbol{\theta};\boldsymbol{\rho}), \end{aligned} \tag{4.50}$$

where χ is the truncated distribution from (4.47). Notice that problem (4.50) can be viewed as a deterministic optimization problem with respect to $\boldsymbol{\rho}$. Then, we can use well-established results from optimization theory and rewrite (4.50) as [61]

$$\begin{aligned} \min_{\boldsymbol{\rho}} \Bigg(& \mathbb{E}\big[J\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big)\big] + \zeta \sum_{i,k} \big(\mathbb{E}\big[g_i^k(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta}))\big]\big)^+ \\ & + \zeta \sum_{j,k} \big|\mathbb{E}\big[h_j^k(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big]\big| \Bigg) \\ \text{s.t. } & \boldsymbol{\theta} \sim \chi(\boldsymbol{\theta};\boldsymbol{\rho}), \end{aligned} \tag{4.51}$$

where $(f)^+ := \max(0, f)$ for any function f and $\zeta > 0$ is an external parameter which controls how the severity of constraint violation is penalized. As shown in [61], a (local) solution to (4.50) is also a (local) minimizer to (4.51) under mild assumptions, when the external parameter satisfies $\zeta > \zeta_*$, for some $\zeta_* > 0$. This is because the corresponding barrier functions are *exact*. This contrasts with schemes where non-exact penalty functions are used (such as the quadratic barrier function [61]) where ζ above has to approach infinity in order to obtain an equivalent solution to (4.50).

One difficutly with solving (4.51) is that the utilized penalty terms are non-differentiable

at zero. To get around this issue, we observe that $(\mathbb{E}[f])^+ \leq \mathbb{E}[(f)^+]$ and $|\mathbb{E}[f]| \leq \mathbb{E}[|f|]$ for any (integrable) function f. Based on this inequality, we will thus be minimizing the upper bound of (4.51):

$$\begin{aligned} \min_{\boldsymbol{\rho}} \Bigg(\mathbb{E}\Bigg[& J\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big) + \zeta \sum_{i,k} \big(g_i^k(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta}))\big)^+ \\ & + \zeta \sum_{j,k} \big|h_j^k(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta})\big| \Bigg] \Bigg) \\ \text{s.t. } & \boldsymbol{\theta} \sim \chi(\boldsymbol{\theta};\boldsymbol{\rho}). \end{aligned} \tag{4.52}$$

Now, problem (4.52) can be solved as suggested by sections 4.3.1 and 4.3.2. In particular, we can apply the shape function $S(\cdot)$ and logarithmic transform from eq. (4.41) and use the update formulas (4.45), (4.48).

4.3.3 Algorithm

We now introduce our constrained sampling-based controller, as summarized in Algorithm 6. Given the constrained optimization problem of the form (4.49), we first initialize the parameters $\boldsymbol{\rho}^{(0)}$ and penalty coefficient ζ. At every iteration m, N control parameter trajectories are sampled from the constrained distribution $\chi(\cdot)$. The state trajectories are obtained from propagating the discrete-time dynamics forward. The constraint-penalized cost for each trajectory is calculated by adding the corresponding penalty terms to the original cost function. The parameters are updated using the Monte-Carlo approximation of a gradient ascent scheme.

As commonly done in constraint optimization, we sequentially increase the penalty coefficient ζ until we converge to a set of parameters $\boldsymbol{\rho}$ that satisfy the constraints. Note that having a high penalty term from the beginning usually makes the problem highly nonlinear and poses difficulties for optimizers, especially when the solution is near the constraint boundaries [61]. Similarly, numerical issues arise when we let the optimizer approach highly infeasible regions for a small ζ, and thus a stopping criterion has to be imposed

Algorithm 6 Constrained sampling-based controller

Require: Initial parameters $\boldsymbol{\rho}_0$, initial penalty coefficient ζ, constraint violation threshold ξ, system dynamics f, inequality constraint g, equality constraint h, cost function J, sample size N, step size α, penalty increase ratio β

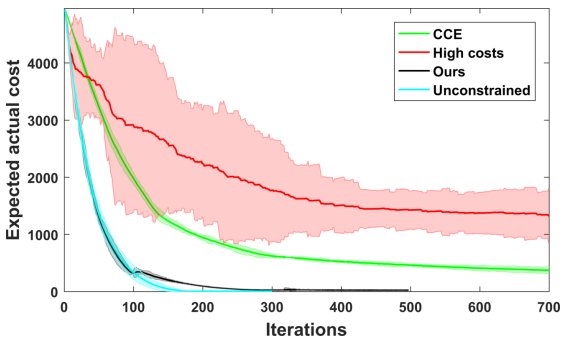
Initialize x^0
Set $i = 1$
while constraints not satisfied **do**
    while $\boldsymbol{\rho}$ not converged **do**
        for n = 1:N **do**
            Sample control parameter trajectories $\boldsymbol{\theta}_n \sim \boldsymbol{\chi}(\boldsymbol{\theta}; \boldsymbol{\rho}_{i-1})$
            for k = 1:H **do**
                Compute state trajectories $x_n^k = f(x_n^{k-1}, u_n^{k-1}(x_n^{k-1}; \theta_n^{k-1}))$;
            end for
            Calculate penalized cost $\tilde{J}_n(\zeta)$
        end for
        Update parameters $\boldsymbol{\rho}^{(m)} = \boldsymbol{\rho}^{(m-1)} + \alpha \Big[\sum_{n=1}^{N} \frac{S(\tilde{J}_n)(T(\boldsymbol{\theta}_n) - \frac{1}{N}\sum_{n=1}^{N} T(\boldsymbol{\theta}_n))}{\sum_{n=1}^{N} S(\tilde{J}_n)} \Big]$
        if any constraint exceeds ξ **then**
            Break
        end if
    end while
    Increase penalty coefficient $\zeta = \beta\zeta$
end while

followed by an increase in the value of ζ.

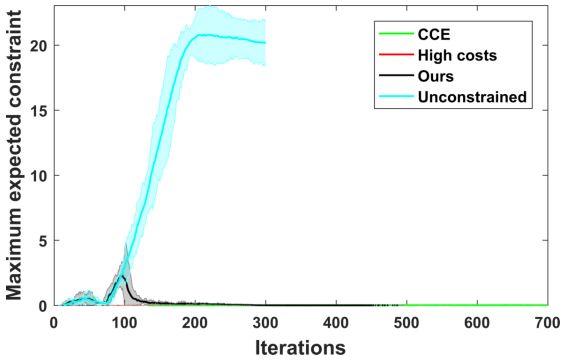
4.3.4 Simulations

We compare our proposed scheme against the constrained cross entropy (CCE) method [62] and an algorithm similar to STOMP [60] on a cart pole and quadcopter in simulation. The latter algorithm is essentially a path integral control scheme, in which each infeasible rollout will receive a cost equal to 10^4. This will be denoted by "High costs" in our figures. Control constraints for these algorithms were handled via clamping. We will also include the results of the path integral algorithm applied on the uncostrained problem of the cartpole simulation (denoted by "Unconstrained").

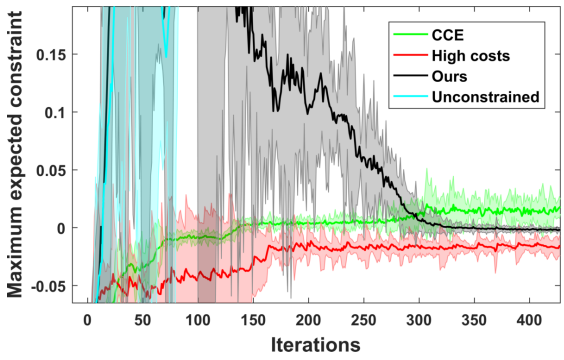
All the simulations are run in Matlab with sample size of 100 rollouts and elite set size of 25 for CCE. The time steps are 0.01 sec and the task horizon is 200 steps. For the cost plots, we ran all algorithms 4 times and plot the mean and the 97% confidence region ($\pm 3\sigma$



(a) Expected cost for cartpole with a $\pm 3\sigma$ confidence interval.



(b) Maximum expected constraints for cartpole with a $\pm 3\sigma$ confidence interval.



(c) Zoomed-in version of Figure 4.9b.

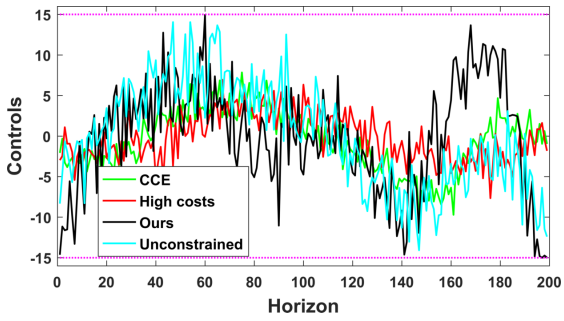


Figure 4.10: Cartpole controls for a single run of the algorithms.

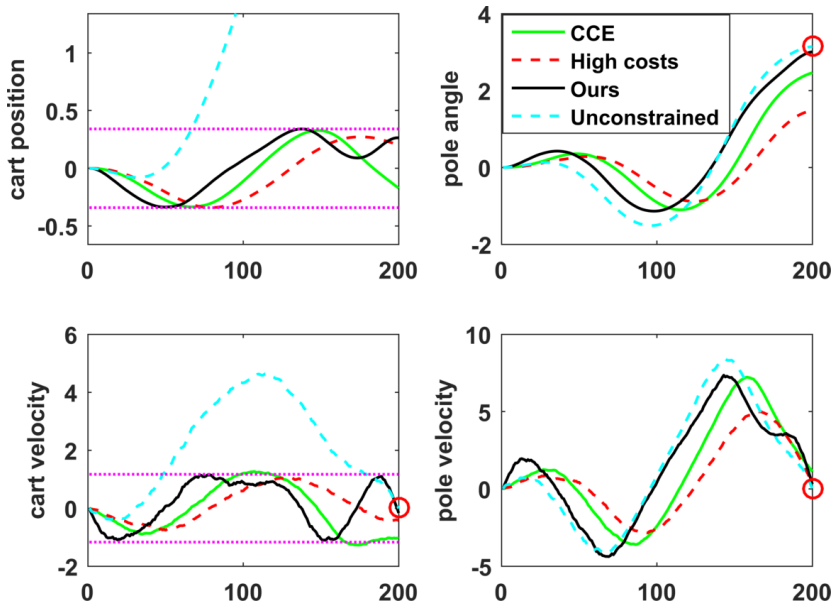


Figure 4.11: Cartpole states for a single run of the algorithms.

in shaded region).For our algorithm, we will consider the truncated Gaussian distribution for the parameter distribution $\chi(\cdot)$. We will also let $\theta^k \equiv u^k$ and only update the mean of the distribution ($\boldsymbol{\rho} = \mu$ of $\chi(\cdot), T(\boldsymbol{\theta}) = \boldsymbol{\theta}$), although the proposed algorithm is applicable to arbitrary distributions in the exponential family and all parameters of the distribution can be updated.

Cartpole

First, we consider the task of a cartpole swing up. The dynamics of the system can be found in [63]. Our initial and target state will be respectively $[0, 0, 0, 0]$, $[-, \pi, 0, 0]$, where the state vector consists of the position of the cart, pole angle, velocity of the cart and pole velocity (note that we do not set a target state for the cart position). The velocity of the pole was influenced by Gaussian noise of variance 0.01. Moreover, the mass of the cart was set to 1 kg, the mass of the pole to 0.1 kg and its length to 0.5 m.

Regarding constraints, we imposed that the position and velocity of the cart satisfy $|\mathsf{x}^k| \leq 0.34\ m$ and $|v^k| \leq 1.17\ m/s$ for all k respectively. For the controls we considered the box constraints $|u^k| \leq 15$. The results of this simulated example can be found in figures

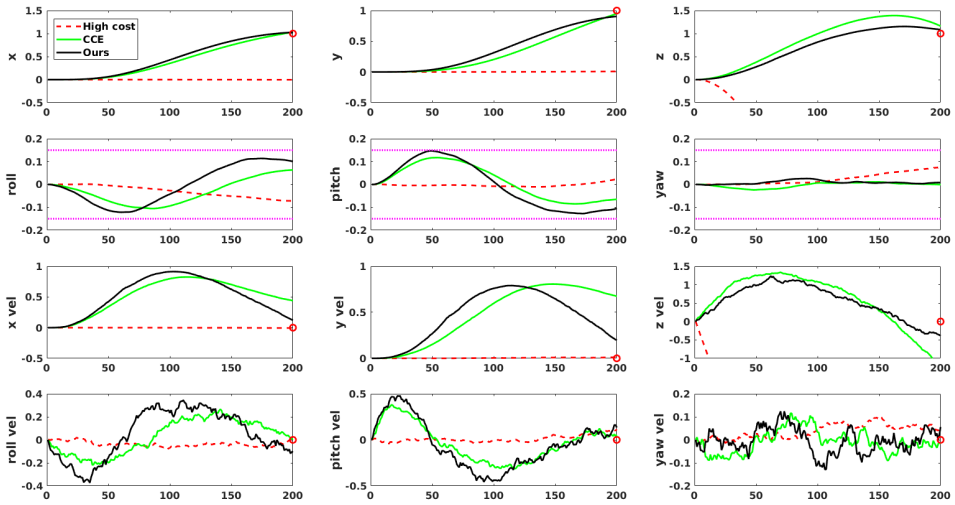


Figure 4.12: Quadcopter states with angular constraints (magenta dotted lines).

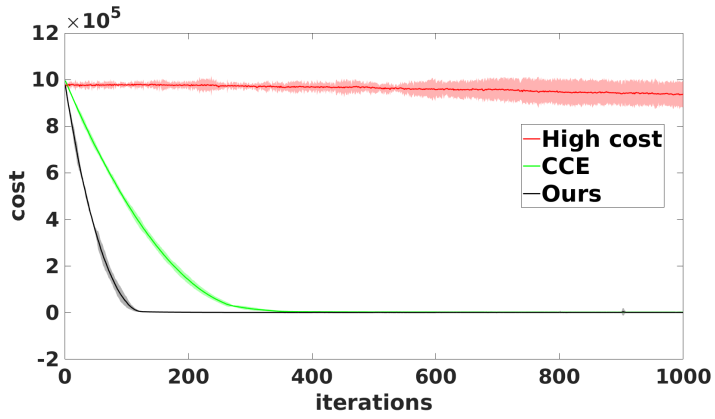


Figure 4.13: Quadcopter cost with a $\pm 3\sigma$ confidence interval.

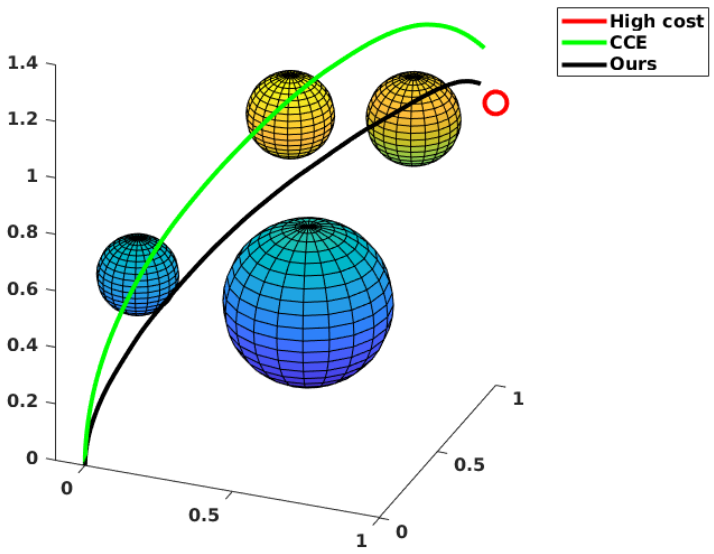


Figure 4.14: Quadcopter state trajectory plot in 3D. The green trajectory denotes the CCE method, and the black trajectory denotes our method. The spheres are the obstacles (nonlinear state constraints).

1-5, where we have let all algorithms run for 700 iterations, unless they converged earlier.

We observe in Fig. 4.9a that our algorithm and CCE clearly outperform the naive penalization approach ("High costs") within path integral. Moreover, our algorithm outperforms CCE in terms of quality of solutions, while the two methods are comparable in terms of feasibility. In particular, as shown in Figures 4.9b, our method initially moves within the infeasible region, which eventually gives a better solution. Furthermore, our method typically gives control inputs which are closer to their boundaries, as depicted in Figure 4.10.

Quadcopter

For the quadcopter, we consider the task of flying from an initial location of $[0, 0, 0]$ meters to a target location of $[1, 1, 1]$ meters. The dynamics of the system can be found in [64]. We imposed nonlinear state constraints on the quadcopter in the form of four obstacles. Additionally, the quadcopter angles are subject to a maximum angle constraint of 0.15

radians.

Figure 4.12 compares the states of the quadcopter and demonstrates that our algorithm and CCE again clearly outperform the high cost approach. Additionally, compared to CCE, our method provides a better solution and converges faster. The superior convergence can be observed in figure 4.13, where our method converges in $1/3$ of the iterations of CCE. Additionally, in figure 4.14, we can observe that our algorithm provides a more direct trajectory that stays close to the obstacles while satisfying the constraints, whereas the optimal trajectory from CCE avoids the obstacles by going above all of them.

4.4 Path Integral Control on Lie groups

Now we are interested in developing a parameterization-free version of PI control for systems evolving on nonlinear configuration spaces. Towards this goal, we employ local mappings between paths on the manifold, and associated trajectories on the Lie algebra. The vector space structure of the latter allows for a derivation analogous to the Euclidean case [16]. Our main theoretical result lies in expressing the optimal controls through the cost of stochastic paths on the Lie group. This observation has also implementation-related implications. In fact, since trajectories can be sampled in a parameterization-free manner, we avoid restrictions associated with the local validity of charts. Therefore, our approach can be viewed as a generalization of the original, Euclidean-based method in [1]. Lastly, notice that in contrast to recent control algorithms on Lie groups (e.g., [65], [66], [67]), our method handles systems with inherent stochasticity.

We will make use of the following lemma:

Lemma 5. *Let G be a Lie group, with $\mathfrak{g}$ denoting its Lie algebra. Let also $y(t)$ define a flow on G that solves the following geometric ODE:*

$$\dot{y}(t) = y(t)\mathsf{f}(t, y(t)), \quad y(t_0) = y_0, \tag{4.53}$$

where $t \geq t_0$ and $\mathsf{f} : \mathbb{R} \times G \to \mathfrak{g}$. Then, for sufficiently small τ, the solution of (4.53) *can be written as*

$$y(t) = y_0 \exp_G(\Theta(t)), \quad \forall t \in [t_0, \tau], \quad \textit{with } \Theta(t) \in \mathfrak{g} \textit{ satisfying} \tag{4.54}$$

$$\dot{\Theta}(t) = \mathrm{dexp}^{-1}_{-\Theta(t)}(\mathsf{f}(t, y_0 \exp_G(\Theta(t)))), \quad \Theta(t_0) = 0. \tag{4.55}$$

A proof can be found in [24, page 38]. We consider systems evolving on the tangent

bundle of a Lie group, G. Since we can always find a one-to-one correspondence between elements of TG and $G \times \mathfrak{g}$, we will represent our state as $x := (g, \xi) \in G \times \mathfrak{g}$. For practical applications, one can think of g as the *pose* of the system, and ξ as its *body-fixed velocity*.

Next, the running cost we consider is $\mathcal{L} = q(s, x(s)) + \frac{1}{2}u(s)^\top Ru(s)$ and the dynamics are assumed to be given by the following set of generic equations:

$$\dot{g}(t) = g(t)\xi(t), \tag{4.56a}$$

$$\mathrm{d}(\xi^\vee)(t) = f(t, x(t))\mathrm{d}t + B(t, g(t))(u(t)\mathrm{d}t + \sigma \mathrm{d}w(t)), \tag{4.56b}$$

$$g(t_0) = g_0, \quad \xi(t_0) = \xi_0, \quad \text{a.s.} \tag{4.56c}$$

We use $g(t)\xi(t)$ as shorthand notation for the vector field $\mathsf{D}_e L_{g(t)}\xi(t)$. We also know that the value function will satisfy $V(t, x_t) = \min_{u(t\to r)} \mathbb{E}[\int_t^r \big(q(s, x_s) + \frac{1}{2}u_s^\top Ru_s\big)\mathrm{d}s + V(r, x_r))\big|\mathcal{F}_t]$, where $\mathcal{F}_t$ is the filtration corresponding to w. Notice that one can pick t, r close enough to a time instant $t_n \in [t_0, t_f)$, such that Lemma 5 and equation (4.56a) imply

$$g_t = g_{t_n} \exp_G(\Theta_t^{t_n}) \quad \text{and} \quad g_r = g_{t_n} \exp_G(\Theta_r^{t_n}), \tag{4.57}$$

where $\Theta_s^{t_n} \equiv \Theta^{t_n}(s) \in \mathfrak{g}$ satisfies for all $s \in [t_n, t_{n+1}]$:

$$\begin{gathered}(\dot{\Theta}^{t_n}(s))^\vee = \rho(s, \Theta^{t_n}(s), \xi(s)), \quad \text{with,} \quad \rho(s, \Theta_s^{t_n}, \xi_s) := (\mathrm{dexp}^{-1}_{-\Theta_s^{t_n}}(\xi_s))^\vee \\ \text{and} \quad \Theta^{t_n}(t_n) = 0.\end{gathered} \tag{4.58}$$

Notice that all we did is rewrite the flow on G via Lemma 5, assuming that the time instances t, r satisfy: $t_n \le t < r \le t_{n+1}$. Here, t_{n+1} denotes the boundary for which Lemma 5 can be applied. Moreover, we use a superscript on Θ to indicate the starting time instant for the corresponding transformations.

Using the above transformations and Itô's Lemma, we can show that the optimal control

at time instant t and value function satisfy respectively (see [68] for more details):

$$u_t^* = -R^{-1}B(t, g_{t_n}\exp_G(\Theta_t^{t_n}))^\top \partial_{(\xi^\vee)}V(t, g_{t_n}\exp_G(\Theta_t^{t_n}), \xi_t). \tag{4.59}$$

$$\begin{aligned}-\partial_t V(t, g_{t_n}\exp_G(\Theta_t^{t_n}), \xi_t) = q(t, g_{t_n}\exp_G(\Theta_t^{t_n}), \xi_t) + (\mathcal{R}_{g_{t_n}}V)(t, \Theta_t^{t_n}, \xi_t) - \\ \frac{1}{2}(\partial_{(\xi^\vee)}V)^\top B(t, g_{t_n}\exp_G(\Theta_t^{t_n}))R^{-1}B(t, g_{t_n}\exp_G(\Theta_t^{t_n}))^\top \partial_{(\xi^\vee)}V(t, g_{t_n}\exp_G(\Theta_t^{t_n}), \xi_t),\end{aligned} \tag{4.60}$$

with $\mathcal{R}_{g_{t_n}}$ properly defined [68] and $t_n \leq t \leq t_{n+1}$. One can immediately see the similarity of (4.60) to the Hamilton-Jacobi-Bellman (HJB) equation from standard stochastic optimal control theory [1]. Notice, however, that eq. (4.60) includes derivatives of the value function with respect to the (artificial) state variable Θ^{t_n}. Moreover, recall that we have used a pullback of $V(\cdot)$ under $(g_{t_n}\exp_G(\cdot))$ which holds for $t \in [t_n, t_{n+1}]$. Therefore, equation (4.60) will only be locally valid. In other words, given any *pose* element of a trajectory, $g_{t_i} \in G$, we will have a corresponding (localized) HJB equation analogous to (4.60), with $v(t, g_{t_i}\exp_G(\Theta_{t_{i+1}}^{t_i}), \xi_{t_{i+1}})$ as the terminal condition and $v(t_f, g_{t_f}, \xi_{t_f}) = \phi(g_{t_f}, \xi_{t_f})$.

Similar to Euclidean Path Integral (PI) control (see, e.g., [1], [16]), we introduce the *desirability function* $\Psi : \mathbb{R} \times G \times \mathfrak{g} \to \mathbb{R}$ as $\Psi(t, x) := \exp(-\frac{1}{\lambda}V(t, x))$, where λ is a positive real constant, and $\exp : \mathbb{R} \to \mathbb{R}$ denotes the standard exponential function. We will also impose the following constraint between noise intensity and control authority: $\lambda B_t R^{-1} B_t^\top = B_t \sigma\sigma^\top B_t^\top$. Then using the mappings in (4.58) and (4.60), we can show that the optimal controls and desirability satisfy (for more details refer to [68]):

$$u_t^* = \lambda R^{-1}B(t, g_{t_n}\exp_G(\Theta_t^{t_n}))^\top \frac{\partial_{(\xi^\vee)}\Psi(t, g_{t_n}\exp_G(\Theta_t^{t_n}), \xi_t)}{\Psi(t, g_{t_n}\exp_G(\Theta_t^{t_n}), \xi_t)}. \tag{4.61}$$

$$
\begin{aligned}
-\partial_t \Psi(t, & g_{t_n} \exp_G(\Theta_t^{t_n}), \xi_t) = \\
& (\mathcal{R}_{g_{t_n}} \Psi)(t, \Theta_t^{t_n}, \xi_t) - \tfrac{1}{\lambda} L(t, g_{t_n} \exp_G(\Theta_t^{t_n}), \xi_t) \Psi(t, g_{t_n} \exp_G(\Theta_t^{t_n}), \xi_t),
\end{aligned}
\tag{4.62}
$$

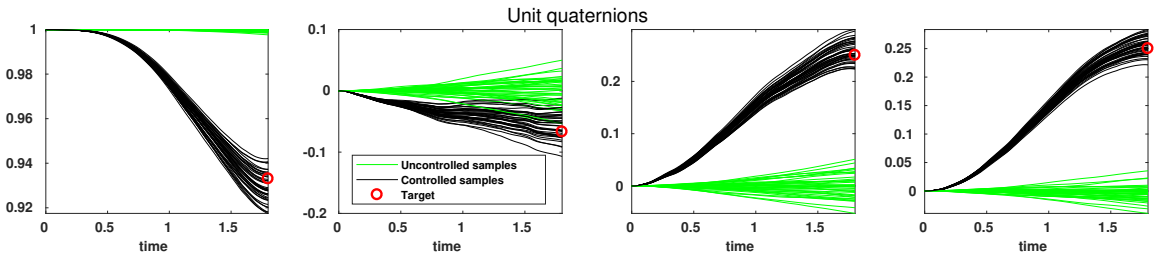
$$
\Psi(t, x_t) = \mathbb{E}\big[\exp\big(-\tfrac{1}{\lambda} S_t \big) \big| \mathcal{F}_t \big], \tag{4.63}
$$

with $S_t = S_t(t \to t_f, x(t \to t_f)) := \int_t^{t_f} L(s, x_s) \mathrm{d}s + \phi(x_{t_f})$. Equation (4.63) is the main theoretical result of this work. It gives a closed form expression for the desirability function, Ψ, for systems with nonlinear configuration spaces and dynamics given by (4.56). Specifically, Ψ is fully determined by the cost of the uncontrolled trajectories on TG. This accords with the corresponding expression we get when flat spaces are considered [1]. Hence, we have proven that the same interpretation carries over to non-Euclidean systems.

This approach clearly shows that both the mathematical derivation, and the algorithmic development of PI control can avoid the use of charts on G. Hence, when approximating eq. (4.63), we can sample paths in a parameterization-free manner, and thus eliminate numerical instabilities occurred by switching charts (i.e., by recursively propagating (4.56b) and (4.58) for each $[t_i, t_{i+1}]$, and then applying the group exponential map.

Simulations

We consider the model of a stochastic rigid satellite whose state evolves on the tangent bundle $TSO(3)$. To numerically apply the results, we discretize the dynamics and cost/desirability functions [68]. Fig. 4.15 depicts 40 sampled trajectories under the obtained controls and also makes a comparison with trajectories sampled from uncontrolled dynamics.



(a) Attitude trajectory over time. A unit quaternion representation is used for illustration.

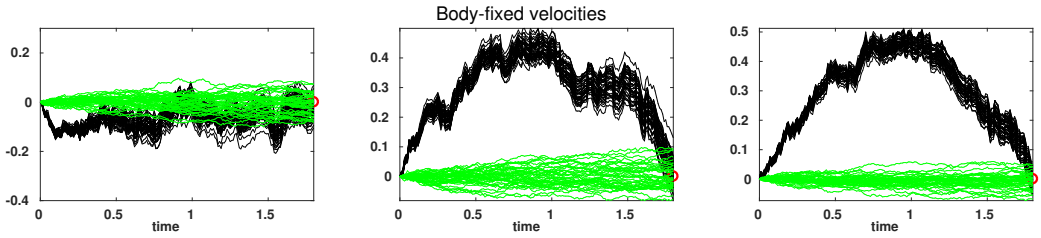


Figure 4.15: Sampled trajectories over the time horizon - part (a) shows a unit quaternion representation of the rotation matrices & part (b) displays the body-fixed velocities. Green lines indicate uncontrolled trajectories, while black lines correspond to the controls generated by PI after 40 iterations.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

The primary objective of this work was to develop methods and theoretical results aimed at constructing scalable control algorithms for deterministic and stochastic systems. We investigated how different uncertainty quantification methods can be employed to model stochasticity and be utilized to build generic control frameworks. State and control constraints were also handled by our trajectory optimizers, via multiplier methods and optimality conditions. Moreover, we showed that optimization-based schemes can be designed for non-Euclidean systems, by using tools from differential geometry, Lie group theory and infinite-dimensional stochastic calculus.

This thesis investigated many different research directions towards developing numerical control schemes for a variety of systems. While significant connections were made among the explored fields, many extensions are yet to be made. We name a few here:

Constrained trajectory optimization: We derived a few methods for constrained control based on Differential Dynamic Programming, multiplier methods and the KKT conditions. An important question is whether a convergence analysis can be developed for these schemes. Furthermore, whether it is possible to have an elegant approach for receding horizon constrained control of discrete systems.

Polynomial Chaos-based control: Polynomial Chaos was one of the methods used to quantify uncertainty in stochastic systems. Although we showed successful numerical results, we did not discuss some major limitations of Polynomial Chaos theory; i.e., inaccurate long-term predictions and lack of scalability with respect to the number of uncertain parameters [5]. Several methods have been proposed to account for these limitations [69]. It will be interesting to see how these extensions can be incorporated within our control algorithms.

Gaussian Processes-based control: Gaussian processes were used in this thesis to account for non-parametric uncertainty within control methods. However, different data-based methodologies share similar probabilistic representations, such as deep learning techniques [70]. The predictive power of the latter approaches might be superior in certain settings, and thus an important extension is using them within dynamic programming-based frameworks.

Trajectory optimization on Lie groups: We derived an optimal control algorithm for deterministic systems evolving on smooth manifolds (specifically, Lie groups), for which numerical results and an extensive convergence analysis was provided. Some interesting extensions include handling state and control constraints, as well as incorporating probabilistic representations on Lie groups [12].

Sampling control of stochastic partial differential equations: We developed an information theoretic approach for control of infinite-dimensional systems. Some practical and theoretical extensions involve using this methodology for actuator placements (we have only considered fixed actuators in this thesis), and incorporating machine learning methods for propagating the state of an unknown field [71].

REFERENCES

[1] H. J. Kappen, "Linear theory for control of nonlinear stochastic systems," *Physical review letters*, vol. 95, no. 20, pp. 200–201, 2005.

[2] M. Deisenroth and C. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *International Conference on Machine Learning, ICML 2011*, Omnipress, 2011.

[3] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*. Elsevier, 1970, ISBN: 0-444-00070-4.

[4] N. Wiener, "The homogeneous chaos," *American Journal of Mathematics*, pp. 897–936, 1938.

[5] D. Xiu and G. E. Karniadakis, "The Wiener–Askey polynomial chaos for stochastic differential equations," *SIAM Journal on scientific computing*, vol. 24, no. 2, pp. 619–644, 2002.

[6] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun, "Geometric, variational integrators for computer animation," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006.

[7] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica 2001*, vol. 10, pp. 357–514, 2001.

[8] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2219–2225.

[9] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005, ISBN: 026218253X.

[10] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd. New York: Springer, 2006.

[11] E. Zhou and J. Hu, "Gradient-based adaptive stochastic search for non-differentiable optimization," *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1818–1832, 2014.

[12] G. S. Chirikjian, *Stochastic models, Information Theory, and Lie Groups*. Birkhäuser, 2009.

[13] G. Da Prato and J. Zabczyk, *Stochastic Equations in Infinite Dimensions*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2008, ISBN: 9780521059800.

[14] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," *ICRA 2014*,

[15] S. E. Shreve, *Stochastic calculus for finance II: Continuous-time models*. Springer Science & Business Media, 2004, vol. 11.

[16] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, no. Nov, pp. 3137–3181, 2010.

[17] E. A. Theodorou, "Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations," *Entropy*, vol. 17, no. 5, pp. 3352–3375, 2015.

[18] E. G. Birgin, R. A. Castillo, and J. M. Martínez, "Numerical comparison of augmented lagrangian algorithms for nonconvex problems," *Computational Optimization and Applications*, vol. 31, no. 1, pp. 31–55, 2005.

[19] D. Xiu, *Numerical methods for stochastic computations : a spectral method approach*. Princeton, N.J. Princeton University Press, 2010.

[20] F. Bullo and A. D. Lewis, *Geometric control of mechanical systems. modeling, analysis and design for simple mechanical control*, 2004.

[21] J. Gallier and J. Quaintance, *Notes on Differential Geometry and Lie Groups*. 2016.

[22] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ: Princeton University Press, 2008, pp. xvi+224, ISBN: 978-0-691-13298-3.

[23] R. Mahony and J. H. Manton, "The geometry of the newton method on non-compact lie groups," *Journal of Global Optimization*, vol. 23, no. 3, pp. 309–327, 2002.

[24] A. Iserles, H. Z. Munthe-Kaas, S. P. Nrsett, and A. Zanna, "Lie-group methods," *ACTA NUMERICA*, pp. 215–365, 2000.

[25] G. H. Golub and J. H. Welsch, "Calculation of gauss quadrature rules," *Mathematics of computation*, vol. 23, no. 106, pp. 221–230, 1969.

[26] E. Johnson, J. Schultz, and T. Murphey, "Structured linearization of discrete mechanical systems for analysis and optimal control," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 140–152, 2015.

[27] M. Leok, "An overview of lie group variational integrators and their applications to optimal control."

[28] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation.," in *Robotics: Science and Systems*, 2017.

[29] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 695–702.

[30] F. S. Hover and M. S. Triantafyllou, "Application of polynomial chaos in stability and control," *Automatica*, vol. 42, no. 5, pp. 789–795, 2006.

[31] J. Fisher and R. Bhattacharya, "Linear quadratic regulation of systems with stochastic parameter uncertainties," *Automatica*, vol. 45, no. 12, pp. 2831–2841, 2009.

[32] G. I. Boutselis, G. De La Torre, and E. A. Theodorou, "Stochastic optimal control using polynomial chaos variational integrators," in *2016 American Control Conference (ACC)*, 2016, pp. 6586–6591.

[33] G. I. Boutselis and E. A. Theodorou, "Spectral variational integrators for trajectory optimization under parametric uncertainties and stochastic disturbances," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 2016–2022.

[34] G. I. Boutselis, Y. Pan, and E. A. Theodorou, "Numerical trajectory optimization for stochastic mechanical systems (to appear)," *SIAM Journal on scientific computing*, 2019.

[35] M. Eldred and J. Burkardt, "Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification," *American Institute of Aeronautics and Astronautics*, 2009.

[36] L.-Z. Liao, *Global Convergence of Differential Dynamic Programming and Newton's Method for Discrete-time Optimal Control*. Technical report, 1996.

[37] L. Z. Liao and C. A. Shoemaker, "Convergence in unconstrained discrete-time differential dynamic programming," *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991.

[38] J. A. Paulson and A. Mesbah, "An efficient method for stochastic optimal control with joint chance constraints for nonlinear systems," *International Journal of Robust and Nonlinear Control*, vol. 29, no. 15, pp. 5017–5037, 2019.

[39] C. Rasmussen and M. Kuss, "Gaussian processes in reinforcement learning," in *Advances in Neural Information Processing Systems 16*, Max-Planck-Gesellschaft, Cambridge, MA, USA: MIT Press, Jun. 2004, pp. 751–759.

[40] Y. Pan, G. I. Boutselis, and E. A. Theodorou, "Efficient reinforcement learning via probabilistic trajectory optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5459–5474, 2018.

[41] G. I. Boutselis and E. Theodorou, "Differential dynamic programming on lie groups: Derivation, convergence analysis and numerical results," *arXiv preprint arXiv:1809.07883*, 2018.

[42] V. S Varadarajan, *Lie groups, Lie algebras, and their representations*. Englewood Cliffs, N.J. : Prentice-Hall, 1974.

[43] A. Astolfi, *Optimization lecture notes*. Imperial College London, 2006.

[44] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 4906–4913.

[45] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033.

[46] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[47] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[48] G. Da Prato and J. Zabczyk, *Stochastic Equations in Infinite Dimensions*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2014, ISBN: 9780521385299.

[49] G. Fabbri, F. Gozzi, and A. Swiech, *Stochastic Optimal Control in Infinite Dimensions - Dynamic Programming and HJB Equations*, ser. Probability Theory and Stochastic Modelling 82. Springer, Jan. 2017, OS.

[50] T. E. Duncan, B Maslowski, and B. Pasik-Duncan, "Ergodic boundary/point control of stochastic semilinear systems," *SIAM journal on control and optimization*, vol. 36, no. 3, pp. 1020–1047, 1998.

[51] G. J. Lord, C. E. Powell, and T. Shardlow, *An Introduction to Computational Stochastic PDEs*, ser. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2014.

[52] G. Da Prato, A. Debussche, and R. Temam, "Stochastic burgers' equation," *Nonlinear Differential Equations and Applications NoDEA*, vol. 1, no. 4, pp. 389–402, 1994.

[53] D.-T. Jeng, "Forced model equation for turbulence," *The Physics of Fluids*, vol. 12, no. 10, pp. 2006–2010, 1969.

[54] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[56] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," 2006.

[57] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[58] L. D. Brown, "Fundamentals of statistical exponential families: With applications in statistical decision theory," Ims, 1986.

[59] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[60] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 4569–4574.

[61] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[62] M. Wen and U. Topcu, "Constrained cross-entropy method for safe reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 7450–7460.

[63] M. P. Deisenroth, *Efficient reinforcement learning using Gaussian processes*. KIT Scientific Publishing, 2010, vol. 9.

[64] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[65] M. Kobilarov, M. Desbrun, J. E. Marsden, and G. S. Sukhatme, "A discrete geometric optimal control framework for systems with symmetries," 2008.

[66] A. Saccon, J. Hauser, and A. P. Aguiar, "Optimal control on lie groups: The projection operator approach," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2230–2245, 2013.

[67] M. B. Kobilarov and J. E. Marsden, "Discrete geometric optimal control on lie groups," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 641–655, 2011.

[68] G. I. Boutselis and E. A. Theodorou, "Path integral control on lie groups," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4990–4995.

[69] X. Wan and G. E. Karniadakis, "An adaptive multi-element generalized polynomial chaos method for stochastic differential equations," *Journal of Computational Physics*, vol. 209, no. 2, pp. 617–642, 2005.

[70] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International Conference on Machine Learning*, 2015, pp. 1861–1869.

[71] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.